Security & Safety Policy

Web Search MCP

Risks, Controls & Customer Policy

Single, universal reference for all customers

Status: Rev 1.0.0 • June 18, 2026

basebox GmbH
Bahnhofplatz 3, 86919 Utting am Ammersee, Germany
+49 (0)8806 9590600 • support@basebox.ai

Confidential / Internal — not for external distribution without review.

Disclaimer

This document is an **internal basebox policy and engineering reference**. It is provided for information and to give customers a clear, consistent basis for deciding how to use the web search capability.

- **Not legally binding.** Nothing in this document constitutes a contract, warranty, service-level commitment, or legal advice. Binding terms are governed solely by the applicable basebox agreement(s) with the customer.
- **Not legal/compliance advice.** References to GDPR, HIPAA, or other regulations are high-level framing only. Customers must confirm their own obligations with their legal counsel / data-protection officer.
- **No guarantee of safety.** As stated throughout, once the open web is in the loop the residual risk can be made small but never zero. This document describes risks and controls; it does not guarantee outcomes.
- **Point-in-time & subject to change.** Roadmap items, third-party vendor terms, and model details are accurate to the best of our knowledge at the date shown and may change without notice.

Revision history & review status

Version	Date	Stage / change	Author(s)	Reviewed by / status
0.1	2026-06-09	Initial problem definition (risk tables, DDG-vs-DokuWiki, customer-facing draft)	Jaro, Markus, Rene	DEV-1115 thread
0.2	2026-06-10	Draft compiled into chaptered policy; research pass (OWASP, NIST, Anthropic, providers); decisions recorded	Jaro	Pending review
1.0	2026-06-18	Reviewed;	Jaro	Version 1.0

Status values: DRAFT → IN REVIEW → APPROVED. Empty rows are reserved for upcoming review cycles.

Contents

01 — Executive Summary

The situation

Web search is one of the most requested capabilities in basebox, and one of the hardest to make safe. The moment an AI assistant can query the open web and read back the results, two things become true at once:

1. **Every customer wants it.** It is frequently the first connector requested.
2. **It cannot be made provably safe.** Once untrusted, attacker-influenceable content from the open web enters the model's context, no vendor — basebox or otherwise — can *guarantee* the system's safety.

A pure risk list always ends at "then don't use it," which serves no one. This document replaces that dead end with a clear policy: the real risks, what basebox does about them, what the customer must own, and the configurations we support.

What basebox already does (defense-in-depth)

- **First-party, single-provider routing.** Search runs through our own web search server to DuckDuckGo only — no third-party search SDK, no arbitrary site fetching, no credentials forwarded, the customer's IP is not exposed to the provider.
- **Text-only ingestion.** HTML is stripped of scripts, styles, and other active content before any text reaches the model; no active content is executed anywhere in the path.
- **Per-call auditing.** Every tool invocation emits an audit event (org, realm, user, tool, provider, success/failure).
- **Tool gating.** Tools are gated per user and per app today.
- **Content-aware framing (per-MCP).** Each first-party MCP server wraps its own tool output in delimiters with a standing instruction that the wrapped text is *data*, never *instructions* - worded for the source (web search frames results as external web content; the wiki connector frames trusted reference data). This keeps the wording accurate for smaller self-hosted models.

What basebox plans to add

- **Host allow-list enforcement** - **on the priority list** (next).
- **Tool-scope reduction** for apps that combine web search with internal-data tools.
- **Per-group** tool visibility and **human-in-the-loop** approval (structures already exist in the data model).

We deliberately do **not** log query content (customers are air-gapped; we do not hold their data) and do **not** ship keyword/output filtering as a security guarantee.

The honest limits

- **Prompt injection cannot be fully solved.** It is mitigated by defense-in-depth and scope reduction, and is **partly a property of the model** (frontier models are explicitly red-teamed against it; smaller self-hosted models are weaker, which makes the architectural controls *more* important).
- **Queries leave the premises.** Anything a user types into a search query is sent to the upstream provider. This is the most realistic everyday risk and is primarily managed by **scope + usage policy**, not by code.

The decision the customer owns

basebox provides the controls and the honest risk picture; the customer decides the risk posture. The recommended, supported baseline is:

1. Enable web search **only in dedicated apps where it is the sole active tool** (eliminates the exfiltration path regardless of model behavior).
2. Limit access to a **defined pilot group**.
3. Add a short **written usage guideline** (e.g. no sensitive/patient data in queries).
4. **Review** after a defined period.

Customers who need stronger guarantees have options (self-hosted metasearch, a curated on-prem corpus, or a third-party provider that shifts — but does not remove — responsibility). These are covered in `chapter 06`.

One-line takeaway

> We can make web search **as safe as is reasonably possible** and be fully transparent about the residual risk — but the open web means the residual risk is never zero, and the customer must consciously accept and co-own it.

02 — Overview & Scope

2.1 Why this document exists

Discussions about enabling web search tend to circle the same point: a list of risks, each reasonable, that collectively read as "this is dangerous." Restated plainly, that list is one thing — *we cannot guarantee the safety of the system once the open web is in the loop.* That is true. But it is only half the picture:

- Every customer wants web search.
- We should not *encourage* unsafe use, but we also should not pretend the capability is something it is not.
- We **can** be transparent about every risk we know or suspect.
- We **can** show exactly what we do to make it as safe as we reasonably can.
- We **can** be honest about alternatives and their trade-offs — including that some options shift responsibility *off* basebox and *onto* the customer.

The goal of this document is to convert that into a **clean, documented policy** we can reuse for every customer instead of re-deriving it each time.

2.2 What "web search" means here

The basebox web search capability is the first-party web search MCP server, which:

- accepts a search query (and optionally fetches a page),
- queries **DuckDuckGo only**,
- extracts **plain text** from results (active content stripped),
- returns that text to the model via the AI service MCP gateway.

The full data flow is described in `chapter 03`.

2.3 Threat model (framing)

We consider the following actors and assets. The detailed enumeration and severity ratings are in `chapter 04`; this is the framing.

Assets to protect

- Sensitive data inside the customer's environment (e.g. internal documents, regulated/personal data).
- The content of user queries (which may themselves be sensitive).
- The integrity of the assistant's behavior and outputs.
- Traceability/accountability of what was searched, by whom.

Adversaries / sources of harm

- A **malicious or compromised web page** returned in search results (indirect prompt injection).
- **Accidental disclosure** by a well-meaning user typing sensitive data into a query.
- The **upstream search provider** as a data recipient.
- A **breach at the provider** exposing previously sent queries.

Trust boundaries

- User → basebox (on-prem): trusted input, but may contain sensitive data.
- basebox → provider (egress): the point where data leaves the premises.

- Provider results → model context: the point where **untrusted** content enters; this is the prompt-injection boundary.

2.4 The two structural problems

Most of the risk reduces to two structural problems that recur throughout the document:

1. **Data leaves the premises** (via the query) — a confidentiality problem, mostly addressable by **scope + policy**.
2. **Untrusted content enters the model** (via results) — an integrity problem, mostly addressable by **defense-in-depth + scope reduction**, and bounded by the model's own injection resistance.

The combination of (2) with *other* enabled tools (e.g. internal-document search) is what turns an integrity problem into a *confidentiality* problem (exfiltration). This is the single most important architectural consideration and is treated explicitly in chapters 04, 05, and 07.

2.5 In scope / out of scope

In scope

- The web search MCP: architecture, data flow, risks, mitigations.
- Supported deployment configurations and customer responsibilities.
- Future search-provider directions.

Out of scope (referenced, not solved here)

- General MCP credential plumbing.
- Policy for non-search MCPs, except where they interact with web search.
- Model selection/procurement, except where it affects injection resistance.

2.6 Relationship to other documents

- Architecture / code: chapter 3.
- Risk detail: chapter 4.
- Controls: chapter 5.
- Provider options: chapter 6.
- Policy & responsibilities: chapter 7.
- External sources: chapter 9.

2.7 Governance & standards alignment

This analysis is grounded in external standards so it is legible to customer security teams and not ad-hoc:

- **OWASP Top 10 for LLM Applications (2025)** provides the *technical taxonomy* for the risks (mapped explicitly in `chapter 04.5`): LLM01 Prompt Injection, LLM02 Sensitive Information Disclosure, LLM05 Improper Output Handling, LLM06 Excessive Agency, LLM09 Misinformation. [OWASP-LLM]
- **NIST AI Risk Management Framework (AI RMF 1.0, 2023)** and its **Generative AI Profile (NIST-AI-600-1, 2024)** provide the *process* backbone (Govern / Map / Measure / Manage). For critical-infrastructure or regulated customers, NIST's in-progress critical-infrastructure profile is also relevant. [NIST-RMF]

- **Agentic-AI / MCP security research** frames the connector-specific risks: the "lethal trifecta" and the structural case for data-layer enforcement [QI-2026] [WILLISON-2025], MCP-specific threat systematizations and empirical attack studies [HOU-2025] [SONG-2025], and the broader large-model/agent safety landscape [MA-2025]. basebox's own internal MCP-security guidance (identity, local access control, network isolation, data-boundary enforcement) is consistent with this literature [BB-MCP-SEC]. These inform `chapters 04-05`.

In short: OWASP for *what can go wrong*, NIST for *how we manage it*, and the agentic-AI literature for *why architectural controls (not the model alone) are the durable answer*.

2.8 Assumptions & non-goals

Assumptions

- Web search is the first-party web search server over streamable-HTTP, with per-request credential handling (no per-user containers by default).
- The catalog/manifest model and AI service gateway behave as documented internally.
- The customer deploys on-prem / in their own environment. Most customers run **air-gapped** on their own hardware with **open-weights** models (e.g. gpt-oss-120b, Mistral Large 3, gpt-oss-20b); cloud deployments use Claude Opus 4.8 (see `chapter 05.6`). basebox has no access to air-gapped customer servers.

Non-goals (explicitly out of this document)

- It is **not legal advice**; GDPR/HIPAA references are high-level framing only.
- It does **not** cover model procurement or selection beyond noting that model choice affects injection resistance.
- It does **not** redesign the MCP credential framework or other connectors.
- It does **not** guarantee safety; it bounds, discloses, and assigns residual risk.

2.9 Glossary

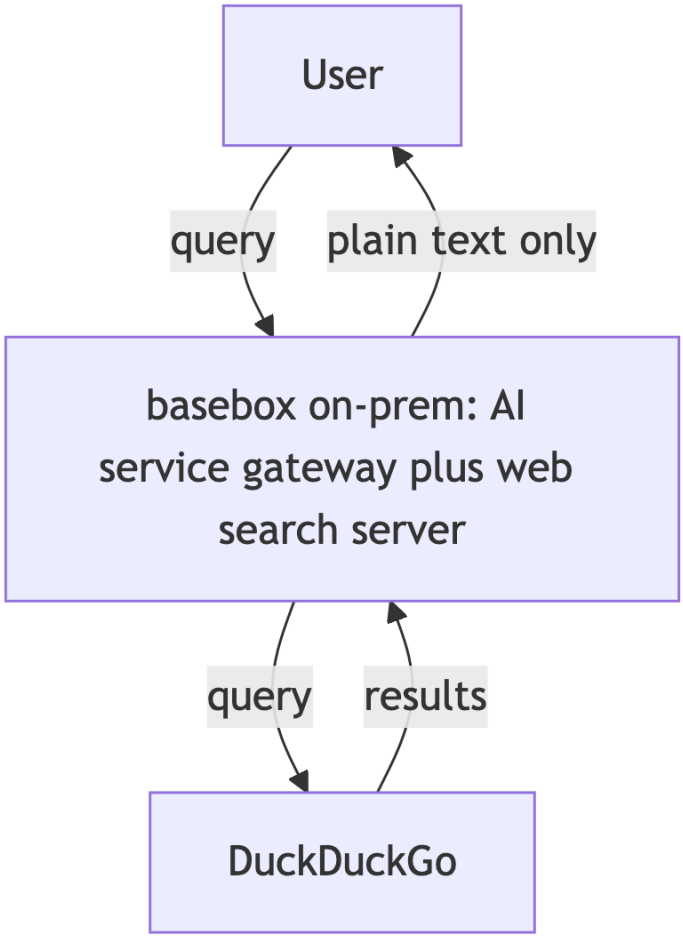
For mixed audiences (this is a single document for technical and security/compliance readers alike).

Term	Meaning
MCP	Model Context Protocol - the standard by which the assistant calls external tools/connectors.
Prompt injection	Untrusted content that the model treats as instructions rather than data, causing unintended behavior.
Indirect prompt injection	Prompt injection delivered via fetched/returned content (e.g. a web page), not typed by the user.
Exfiltration	Unauthorized movement of internal data out of the environment (here, via a follow-up search query).
Query leakage	Sensitive data leaving the premises because it was typed into a search query.
Untrusted-content framing	Wrapping tool results in delimiters + a standing instruction that the content is data, never instructions.
Scope reduction	Restricting which tools can be used after untrusted content enters context.
Allow-list	A fixed set of permitted destinations (here, DuckDuckGo hosts only).
ZDR (Zero Data Retention)	A vendor commitment not to retain submitted data.
DPA (Data Processing Agreement)	Contract governing how a processor handles a controller's personal data.

Term	Meaning
Controller / Processor	GDPR roles: the controller decides why/how data is processed; the processor acts on the controller's behalf.
Residual risk	The risk that remains after controls are applied.
Tier A / sole active tool	The recommended config: web search is the only active tool in an app, removing the exfiltration path.

03 — How Web Search Works

3.1 High-level data flow



Within basebox, the flow is: the AI service receives the tool call, forwards it to the first-party web search server, which queries DuckDuckGo, extracts plain text, and returns it; the AI service places that text in the model's context and audits the call.

Key properties of this path:

- Queries are routed through basebox **on the customer's premises**, not from the user's device.
- **Only DuckDuckGo** is reached - no other search provider, no arbitrary third-party search SDK.
- Only **plain text** reaches the model; active content is stripped.
- The customer's IP is **not** exposed to DuckDuckGo - the egress IP is the basebox MCP server's.
- **No credentials** are forwarded for this connector.

3.2 Components in the path

Component	Responsibility	Notes
AI service MCP gateway	Registers MCP providers from the baked-in catalog, gates tools per user/app, forwards calls, emits audit events	The chat-path entry point
Web search server (first-party)	Performs the DuckDuckGo query, fetches/extracts text, returns a normalized result envelope	In-repo; not a third-party server
DuckDuckGo	Upstream search index	Receives the query text from the basebox egress IP
Model	Consumes the returned text	Injection resistance is partly a model property (see chapters 04 and 05)

3.3 Text-only extraction (no active content)

The web search server extracts text from HTML and explicitly removes active and non-content elements before returning anything to the model. The extractor:

- strips a set of **noise tags** (e.g. `script`, `style`, `noscript`, `iframe`, `svg`, `form`, `nav`, `header`, `footer`, `button`, `input`, …), and
- inserts boundaries around **block tags** so the result is readable text rather than a run-on string.

Consequently, **no JavaScript or other active content is executed** anywhere in the path, and only plain text reaches the model.

> The extractor strips a defined set of noise tags (e.g. `script`, `style`, `noscript`, `iframe`, `svg`, `form`, `nav`, `header`, `footer`, `button`, `input`) and inserts boundaries around block-level elements to produce readable text. Verified 2026-06-10.

3.4 What leaves the premises, and what does not

Item	Egress?	Notes
The **query text**	**Yes**	Sent to DuckDuckGo
The user's **IP address**	No	Only the MCP server's egress IP is seen
Credentials	No	None forwarded for search
Internal documents / app data	No	Not by web search itself — see combined-tools path in `chapter 04`.

This table is the basis for the confidentiality analysis in `chapter 04`: the query is the data-egress channel; everything else stays inside.

3.5 Auditing

Every MCP tool invocation emits an audit event capturing **org, realm, actor(user), tool name, provider, and success/failure**.

> Every MCP tool invocation emits an audit event capturing org, realm, actor (user), tool name, provider, and success/failure. Verified 2026-06-10.

Current gap: the event records the *tool and provider*, not the *query string* or a result reference. Logging the query would enable "who searched *what*" — but **by policy we deliberately do not do this** (customers are air-gapped, we do not hold their query content; see `chapter 05.4` item 6). It would be added only if a majority of customers request it, as a customer-controlled opt-in.

3.6 What this architecture already buys us

Before any additional controls, the design already provides:

- single-provider, first-party routing (no third-party search SDK),
- text-only ingestion (no active-content execution),
- per-call auditing,
- per-user and per-app tool gating.

The remaining gaps and planned controls are the subject of `chapter 05`.

04 — Risk Assessment

4.1 Severity scale

Severity	Meaning
High	Plausible path to disclosure of sensitive/regulated data, or compliance violation
Medium	Real risk, but bounded impact or requires additional conditions
Low	Edge case or low-impact; mostly procedural

Severity reflects **residual** risk given the current architecture (`chapter 03`), before the additional controls planned in `chapter 05`.

Likelihood scale

Likelihood	Meaning
High	Expected to occur in normal use without specific controls
Medium	Plausible; needs some conditions or an attacker effort
Low	Requires unusual conditions or is mitigated by design

Risk = Likelihood x Impact. The register below gives both, plus the **residual risk after the recommended controls** (Tier A isolation + in-place per-MCP framing + planned allow-list, `chapter 05`/`07`).

4.1.1 Risk summary (Likelihood x Impact -> Residual)

Risk	Likelihood	Impact	Inherent	Residual after recommended controls
P1 Prompt injection (alone)	High	Medium	High	Low-Medium (framing + model)
P1b Combine tools exfiltration	Medium	High	**High**	**Low** (Tier A removes the path)
P2 Query leakage	High	High (regulated)	**High**	Medium (scope + usage policy)
P3 Provider exposure	High	Medium	Medium	Low-Medium (first-party routing, DDG policy)
P4 Traceability gap	Medium	Medium	Medium	Medium (accepted by policy; no query logging)

Risk	Likelihood	Impact	Inherent	Residual after recommended controls
P5 Active-content execu-tion	Low	Low	Low	Low (mitigated by design)

Key point: the two **High** inherent risks (P1b, P2) drop the most under the recommended policy - P1b to **Low** via Tier A isolation, P2 to **Medium** via scope + usage policy. This is the quantified basis for the Tier A recommendation.

4.2 Risk register (web search)

The five problem areas below consolidate the technical analysis from the DEV-1115 thread.

Problem 1 - Prompt injection via search results (integrity)

A malicious or compromised page returned in results contains hidden text that the model may treat as instructions rather than data ("indirect prompt injection") [GRESHAKE-2023]. With web search *alone*, the worst case is a misleading or manipulated answer.

Severity: High (because of what it enables when other tools are present — see Problem 1b). Systematic studies confirm this is a general, unsolved class of vulnerability in LLM-integrated applications [LIU-2024] [QI-2026], not specific to any one product or model.

Problem 1b - Exfiltration via combined tools (confidentiality)

If web search is enabled in the **same app** as an internal-data tool (wiki, files, etc.), an injected instruction can try to (a) read internal data via the second tool and (b) place it into a follow-up search query, sending it outside. This converts an integrity problem into a confidentiality breach.

Severity: High. This is the single most important risk and the reason for the "sole active tool" recommendation (`chapter 05/07`).

This pattern is the now-widely-recognized **"lethal trifecta"** — the dangerous combination of (1) access to private data, (2) exposure to untrusted content, and (3) an outbound communication channel; any agent that has all three can be tricked into exfiltrating data [WILLISON-2025] [QI-2026]. It is not hypothetical:

the **EchoLeak** vulnerability in Microsoft 365 Copilot (CVE-2025-32711) demonstrated zero-click data exfiltration triggered purely by a crafted email [KITEWORKS-2026], and peer-reviewed benchmarks of tool-using agents categorize private-data exfiltration as a primary attack class with measurable success rates [ZHAN-2024]. A recent peer-reviewed survey stresses the structural point that model-layer defenses (system prompts, safety training) cannot reliably stop such data-layer attacks — enforcement must be architectural [QI-2026].

Current deployment reality (2026-06-10): the only client-side tool in scope today besides web search is the internal wiki connector (read-only); there are no other tools enabled on the client side. So the *concrete* P1b path today is "web search and wiki access in the same app." Keeping them in **separate apps** (Tier A) eliminates the path now; as more tools are added the same rule applies.

The "lethal trifecta." P1b is the well-documented *lethal trifecta* pattern: an agent that simultaneously (1) has access to private data, (2) is exposed to untrusted content, and (3) can communicate externally is structurally exploitable [WILLISON-2025] [QI-2026]. Web search supplies

(2) and (3); any internal-data tool supplies (1). Peer-reviewed work formalizes this and shows it is a *structural* flaw, not a configuration error [QI-2026], with measured exfiltration rates against tool-using agents (e.g. ~24% for a ReAct GPT-4 agent) [ZHAN-2024]. A real-world instance is **EchoLeak (CVE-2025-32711)** in Microsoft 365 Copilot, where a crafted email triggered data exfiltration with zero user interaction [KITEWORKS-2026].

Problem 2 - Data leakage through search queries (confidentiality)

A user types sensitive content (e.g. a patient name or case detail) into a query; the query leaves the premises to the provider. This is the **most realistic everyday risk**. Free-text query filtering is unreliable; the effective levers are **scope + usage policy**.

Severity: High in regulated environments; Low-Medium elsewhere.

Problem 3 - Data exposure to the search provider (confidentiality)

Even absent user error, every query is seen by the provider, who may log it. basebox routing means the **user's IP is not exposed** (only the server's egress IP) and DuckDuckGo's stated policy is no profiling/no storage of personal data. A breach at the provider could expose previously sent queries.

Severity: Medium (High where a data-processing agreement is legally required).

Problem 4 - Traceability per call (accountability)

Per-call auditing exists (org, realm, user, tool, provider, success/failure) but does **not** record the query string or a result reference, so "who searched *what*" is not currently answerable.

Severity: Medium (compliance-dependent). **By policy we deliberately do not log query content** (air-gapped customers; we do not hold their data - `chapter 05.4` item 6), so this gap is an accepted, documented choice rather than a planned fix.

Problem 5 - Malicious content execution (integrity)

Not a concern in the current design: only plain text is extracted; scripts and active content are stripped before anything reaches the model (`chapter 03.3`).

Severity: Low (mitigated by design).

4.3 Comparison: DuckDuckGo web search vs DokuWiki search

The original DEV-1115 question concerned running two read-only MCPs: web search and DokuWiki. The contrast clarifies that the risk comes from **the open web**, not from "search" as a feature. (Rows continue in the table below.)

#	Risk	DuckDuckGo Web Search	DokuWiki Search	Severity
1	Sensitive data sent outside the org	Query details may be sent to DuckDuckGo's servers	Data stays within the org's own network	High
2	Data stored by a third party	DuckDuckGo may log queries, even if temporarily	No third-party storage	High

#	Risk	DuckDuckGo Web Search	DokuWiki Search	Severity
3	No control over where data travels	Queries cross the public internet	Data never leaves the internal network	Medium
4	Breach via external service	A breach at DuckDuckGo could expose past queries	No external exposure; internal policy applies	Medium
5	Compliance (GDPR / HIPAA)	Sending regulated queries abroad may violate data-protection law absent a data-processing agreement	Fully under org control; easier to stay compliant	High
6	Staff accidentally sharing sensitive data	Easy to include a name or case detail in a query	Same risk, but data stays internal	Low-Medium
7	Encryption in transit	Depends on DuckDuckGo's infrastructure; org has no guarantee	Depends on internal network; org has full control	Medium
8	Prompt injection via returned content	A malicious site in results could carry hidden instructions that manipulate the AI	A compromised wiki page could likewise carry hidden instructions within the session	High

4.4 What the comparison shows

- The DokuWiki column is consistently safer on **confidentiality** rows (1-7): data stays internal. The decisive difference is **egress to the open web**, not the act of searching.
- The two converge on row 8 (**prompt injection**): *any* tool that brings attacker-influenceable text into context carries injection risk. The open web is simply the largest, least-controllable source of such text.

This is why the policy (`chapter 07`) treats web search as the highest-scrutiny connector and why the key control is **isolating it from other tools**, not banning it.

4.5 Mapping to OWASP Top 10 for LLM Applications (2025)

Our risk register aligns with the industry-standard taxonomy [OWASP-LLM]:

Our risk	OWASP 2025
P1 / P1b - prompt injection, combined-tools exfiltration	**LLM01 Prompt Injection** (+ **LLM06 Excessive Agency** for the tool-chaining aspect)
P2 - query leakage	**LLM02 Sensitive Information Disclosure**
P3 - provider exposure	**LLM02** (third-party exposure dimension)
P4 - traceability gap	governance/accountability (cross-cutting)
P5 - active-content execution	**LLM05 Improper Output Handling** (mitigated by text-only)
Manipulated answers (effect of P1)	**LLM09 Misinformation**

Using the OWASP mapping keeps the assessment legible to customer security teams and shows the analysis is grounded in an external standard, not ad-hoc.

4.6 Compliance framing (GDPR / HIPAA) - high-level, not legal advice

Web search queries can carry personal or regulated data out of the environment, which brings data-protection law into scope. This is framing for the conversation, not legal advice (see the non-goals in `chapter 02.8`); customers should confirm with their own counsel/DPO.

GDPR (EU/EEA, and similar regimes). If a query contains personal data, sending it to an external search provider is a *processing* and likely a *transfer* of personal data. Key points:

- The **customer is the data controller**; the search provider is a **processor/sub-processor** (and may also be an independent controller for its own logs) - see `chapter 06.7`. A **Data Processing Agreement (DPA)** and a lawful basis are required.
- **Cross-border transfers** (e.g. to a US provider) need a valid transfer mechanism. basebox's first-party routing limits exposure (no user IP, anonymous queries with DuckDuckGo - `chapter 03`/[DDG-PRIV]), but the **query text still leaves**, so this is not eliminated.
- **Data minimization** (Art. 5): the strongest lever is simply not putting personal data in queries - hence the usage guideline and Tier A scope (`chapter 07`).

HIPAA (US healthcare). If queries could contain Protected Health Information (PHI):

- A general web search provider is typically **not a HIPAA Business Associate** and will not sign a BAA for ad-hoc search; therefore **PHI must not be placed in web search queries**.
- The practical control is the same: **scope + usage policy** (no patient/case detail in queries), reinforced by Tier A isolation so an injected instruction cannot exfiltrate PHI via search.

What this means for the policy. Compliance does not forbid web search; it shapes *how* it is offered: minimize what goes into queries, keep web search isolated (Tier A), prefer first-party or on-prem options for sensitive contexts (`chapter 06`), and document the controller/processor split and any DPA. Audit logging of queries (`chapter 05.4` item 6) is itself a data-protection decision — it improves traceability but stores potentially sensitive data, so it must be a deliberate, documented choice.

05 — Mitigations & Controls

5.1 Guiding principle

Prompt injection cannot be fully solved. The strategy is therefore **defense-in-depth + scope reduction**: layer cheap, robust controls, and shrink what an injected instruction could actually achieve. We prefer controls that reduce *capability* (e.g. isolating tools) over controls that try to *detect* malice (e.g. content filtering), because the latter give false confidence. This matches the consensus in the research literature: no single defense fully prevents prompt injection [GRESHAKE-2023] [LIU-2024], and the durable mitigations are **structural / architectural** rather than prompt-level guardrails [QI-2026] [BEURER-2025] [CAMEL-2025].

5.2 Controls overview

Control	Addresses	Status	Note
First-party single-provider routing	P3	In place	DuckDuckGo only
Text-only extraction (no active content)	P5	In place	HTML noise/active tags stripped
Per-call audit event	P4	In place	tool+provider; not query (gap)
Per-user / per-app tool gating	P1b, P2	Partial	app-scoped today, not per-group
Untrusted-content framing (per-MCP, content-aware)	P1, P1b	In place	implemented at each MCP server; web results framed as external web content, never instructions
Host allow-list enforcement	P1, P3	Priority (next)	infra-level today; enforce in MCP/NetworkPolicy
Tool-scope reduction (no tools after web result)	P1b	Not present	planned; dynamic scoping
Per-group tool visibility	P2	Not present	needs app+group sharing build
Query in audit log	P4	Not planned (policy)	air-gapped; we don't hold query content; opt-in only if majority ask
Human-in-the-loop approval	P1b, P2	Partial	`requires_human_approval` flag exists in DB
Output/keyword filtering of results	P1	Not built (by choice)	easily bypassed; false confidence

5.3 In place today

- **First-party, single-provider routing** to DuckDuckGo (no third-party SDK, no credential forwarding, user IP not exposed). Addresses P3.

- **Text-only extraction**: scripts/active content stripped before the model sees anything. Addresses P5.
- **Per-call auditing**: `mcp.tool.invoke` with org/realm/user/tool/provider/result. Partially addresses P4.
- **Per-user / per-app tool gating** via the tool settings schema. The structures exist; per-app, per-tool enablement is currently disabled in the pilot (RC.1) and needs hardening (e.g. handling upstream tool-list changes). Partially addresses P1b and P2.
- **Untrusted-content framing (per-MCP, content-aware)**: each first-party MCP server wraps its own tool output in explicit delimiters with a standing instruction that the wrapped text is *content/data*, never *instructions*. The framing is deliberately **per-MCP and content-appropriate**, not a blanket gateway wrapper: web search frames results as **external, untrusted web content**, whereas a trusted internal source such as the wiki connector frames its output as **trusted reference data that must still not be interpreted as instructions** (the relevant distinction for injection is *data vs instruction*, not *trusted vs untrusted*). Doing this at the source keeps the wording precise and avoids polluting the prompt of smaller self-hosted models with inaccurate "untrusted" labels. This is the "spotlighting" / data-instruction-separation technique [HINES-2024]. Addresses P1 and P1b.

  The tool gating schema defines tool settings at the per-tool, per-user, and per-app levels (including `is_enabled` and `requires_human_approval` flags). **Nuance:** `requires_human_approval` exists at the **tool** and **per-user** levels but **not** on the per-app settings table — so app-level human approval would need either the user-level flag or a schema addition.

5.4 Planned (recommended order)

Decision (2026-06-10): item 1 (per-MCP content-aware framing) was **committed and is now implemented** (see §5.3); item 2 (host allow-list) is **on the priority list** to follow. Items 3-6 remain planned/optional as noted.

1. **Untrusted-content framing (per-MCP, content-aware) - DONE.** Each first-party MCP server wraps its own tool output in explicit delimiters with a standing instruction that the wrapped text is *data*, never *instructions*, using wording appropriate to the source (external web content for search; trusted-but-data for the wiki connector). Small, high-value; the single best cheap win. This is the "spotlighting" / data-instruction-separation technique, shown to cut indirect prompt-injection success from >50% to <2% in controlled tests [HINES-2024]. **Status: in place.** (P1, P1b)
2. **Host allow-list enforcement** - enforce the DuckDuckGo-only allow-list beyond infra config: at the MCP level (block fetches to other hosts) and/or as a Kubernetes egress `NetworkPolicy` so egress is guaranteed. A `NetworkPolicy` with an `Egress` rule restricting the MCP pod to the DuckDuckGo hosts/ports (plus DNS) makes "DDG-only" a network guarantee rather than a code convention; the Helm charts are the natural place for it. **Status: on the priority list (next after framing).** (P1, P3)
3. **Tool-scope reduction** - after untrusted web content enters context, restrict which tools may still be called (dynamic scoping), closing the exfiltration path even when tools are co-enabled. This is the structural, provable-by-design direction in current research (capability/flow control, secure-by-design agent patterns) [CAMEL-2025] [BEURER-2025]. (P1b)
4. **Per-group tool visibility** - a clean per-group toggle (needs app+group sharing). (P2)
5. **Human-in-the-loop approval** - implement the existing per-tool `requires_human_approval` flag as an interactive gate (present at the tool and per-user levels; add it to per-app settings if app-level approval is wanted). (P1b, P2)
6. **Query auditing (not planned - by policy). Decision (2026-06-10): we will NOT add**

query-content to the audit log by default. It is not our target or policy: customers run **air-gapped** deployments on their own hardware, we do **not** have (or want) access to those servers, and logging query content would store potentially sensitive data we have no business holding. We will add it **only if a majority of customers explicitly request it**, and then only as a customer-controlled, opt-in feature on their side. (P4)

5.5 Deliberately NOT building

- **Output/keyword filtering of results or free-text query filtering** as a security guard. It is trivially bypassed and creates false confidence. If ever offered, it must be labeled **best-effort**, never a guarantee. (P1, P2)

5.6 Model dependency (not a code fix)

Prompt-injection resistance is **partly a property of the model**. Frontier models (e.g. Claude Opus-class) are explicitly trained and red-teamed against injection; a smaller self-hosted model has weaker built-in resistance. With a non-frontier, self-hosted model (e.g. an open-weights model), the prompt-level framing and **scope reduction matter more**, not less. This is a model-selection trade-off, not something we can "fix" in code.

This is consistent with vendor evidence: Anthropic's Transparency Hub reports material *but never complete* injection resistance even for frontier models - e.g. safeguards raising attack-prevention from 71% to 89% (Claude 4), and from 74% to 88% on an earlier model, with the explicit framing that they "aim to continue enhancing" these systems [ANTHROPIC-TRANS]. In other words, even the best models leave residual risk; a weaker self-hosted model leaves more, so the architectural controls (isolation, framing, allow-listing) carry more weight.

Models in scope (2026-06-10). basebox supports whatever the customer's hardware can run. In practice:

Deployment	Model(s)	Injection-resistance note
Air-gapped (customer hardware)	**gpt-oss-120b** (most common); **Mistral Large 3 675B Instruct 2512**; **gpt-oss-20b** (small GPUs)	open-weights models; weaker built-in injection resistance than frontier -> architectural controls matter most here
Cloud	**Claude Opus 4.8**	frontier model, strongest built-in injection resistance, but still not 100%

Implication: most regulated/air-gapped customers run **open-weights** models, so the Tier A isolation + framing + allow-list controls are doing the heavy lifting; we do not rely on the model alone. Model choice is constrained by the customer's hardware, not a free variable. (Internal model-selection and on-prem serving context: [BB-MODELS].)

External guidance on these controls (framing, defense-in-depth, scope reduction, human-in-the-loop, and the "cannot be fully solved" framing) is collected in `chapter 09` and summarized in 5.6.1.

5.6.1 Industry context (this is not vendor-specific)

The model-dependency point is not unique to one vendor. Every major lab uses the same **defense-in-depth + governance** pattern and none claims complete immunity to prompt in-

jection - which is exactly why our architectural controls matter regardless of which model a customer runs.

Lab / model	Governance framework	Defense pattern	Residual risk stated?
Anthropic (Claude)	Responsible Scaling Policy / ASL	Training + classifiers/guards + red teaming	Yes ("aim to continue enhancing") [ANTHROPIC-TRANS] [ANTHROPIC-FABLE]
OpenAI (ChatGPT)	Preparedness Framework (deploy only if <= Medium)	Post-training refusals + Moderation API over inputs/outputs + external red teams	Yes [OPENAI-GPT4O]
Google (Gemini)	Secure AI Framework (SAIF), 6 elements; AI Red Team	Secure-by-default controls + red teaming; aligns with NIST	Yes [GOOGLE-SAIF]
xAI (Grok)	Risk Management Framework	Training + safety monitoring; asks users not to submit personal data	Yes ("no … method … is 100% secure") [XAI-PRIV]

All four broadly **align with NIST AI RMF**, and Google's industry coalition (CoSAI) includes Anthropic among others - i.e. these controls are an emerging industry norm, not a one-vendor opinion. Sources in `chapter 09`.

Implication for the policy: customers may run different models; our controls (isolation/Tier A, framing, allow-list, auditing) are the constant. A stronger model reduces residual injection risk but never to zero; a weaker self-hosted model makes the architectural controls more important. Independent benchmarks of tool-using agents (e.g. AgentDojo, InjecAgent) confirm that no current model is robust to injection on its own, reinforcing the architecture-first approach [DEBENEDETTI-2024] [ZHAN-2024].

5.7 Incident response (when, not if)

Because residual risk is never zero, the policy assumes incidents will eventually occur and defines how to handle them. This aligns with basebox's existing ISO 27001 Incident Response Process [BB-INS-IR] and with the control set that recent agent-security research identifies as non-negotiable — agent-identity authentication, operation-level access control, encryption, **tamper-evident audit logging**, credential vaulting, and zero-trust intake of untrusted content [QI-2026].

TODO: map each to the platform's security runbooks.

Detect

- Audit events (`mcp.tool.invoke`) give per-call visibility (org/realm/user/tool/provider). Note: by policy we do **not** log query content (chapter 05.4 item 6), so detection works at the tool/provider/actor level, not on query text.
- Watch for anomalies: spikes in search volume, or tool-call sequences mixing internal-data tools with search in the same app.

Contain

- **Kill-switch:** disable the web search tool for an app/group/tenant via the per-app/per-user tool gating (5.3). This is the fastest containment lever.
- For a suspected exfiltration attempt, disable web search in any app where it is co-enabled with internal-data tools (or move that app to Tier A).

Notify

- Inform the affected customer per the agreed contacts/SLA; if regulated data may have left the premises, follow the customer's breach-notification obligations (the customer is the data controller - `chapter 06.7`).

Learn

- Record the incident, root cause, and fix; feed it back into this document and the control roadmap (5.4). Update the usage guideline if user behavior was a factor.

  Note: this is a policy-level outline, not a full runbook. The concrete runbook (owners, escalation, timelines) should live with the platform's security operations and be referenced here.

06 — Alternatives & Future Search-Provider Directions

6.1 The responsibility trade-off

The current default (first-party web search server routing to DuckDuckGo) keeps basebox in the loop and minimizes data exposure, but it also keeps **responsibility with basebox**. Some alternatives shift responsibility - and risk - **onto the customer** (e.g. a customer-chosen third-party API), or remove the open-web exposure entirely (e.g. a curated internal corpus). None of these is strictly “better”; each moves the trade-off. This chapter lays out the options so a customer can choose deliberately.

Two axes matter most:

- **Where do queries go?** (own infra / a third party / nowhere external)
- **Who owns the residual risk?** (basebox / the customer / no open-web risk)

6.2 Options matrix

Option	Type	Why consider it	Trade-offs / fit
SearXNG (self-hosted metasearch)	Open-source, self-hosted	**Best fit for airgap/regulated**: we host it, queries do not leave our infra to a third party, aggregates many engines, no profiling, full control	Needs hosting + maintenance; still makes outbound calls to upstream engines *unless* paired with a local index; ops burden. Strong direction for privacy-first customers.
Brave Search API	Paid API	Independent index (not a Bing/Google reseller), privacy-positioned, clean commercial API, predictable	Outbound to Brave; per-query cost; still a third party seeing queries.
Kagi Search API	Paid API	Privacy-first, high-quality results, no ads/tracking, strong reputation	Paid; outbound third party; per-seat/usage cost.
Bing / Azure AI Search (Grounding)	Paid API	Enterprise SLA, Azure-native for Microsoft-shop customers	Microsoft sees queries; not airgap-friendly; cost.
Tavily / You.com / Exa (LLM-native search APIs)	Paid API	Purpose-built for agent/RAG use; return clean ranked snippets ideal for LLMs; less HTML-scraping fragility	Outbound third party; cost; newer vendors.
Perplexity API	Paid API	Returns synthesized answers + citations (less raw-HTML injection surface)	Answer-style is not raw search; outbound third party; cost.
On-prem / curated corpus search	Internal	Ultimate privacy: search only an approved internal index (or a mirrored allow-list of domains)	No open web; build/maintain the index; but **zero injection-from-open-web and zero leakage** - arguably the safest answer for a regulated customer.

6.3 How each option maps to the two axes

Option	Queries go to	Residual-risk owner
Web search server -> DuckDuckGo (current)	DuckDuckGo (via our egress)	basebox
SearXNG (self-hosted)	upstream engines, via our infra	basebox (reduced)
Brave / Kagi / Bing / Tavily / You.com / Exa / Perplexity	the chosen third party	**customer** (their vendor choice)
On-prem / curated corpus	nowhere external	no open-web risk

6.4 Recommended directions

- **Privacy-first / regulated customers:** prioritize **SearXNG (self-hosted)** or a **curated on-prem corpus**. The corpus option removes open-web injection and leakage entirely and is the strongest answer where the open web is not strictly required.
- **Customers who explicitly want a commercial provider:** offer a clearly-named third-party option (e.g. Brave/Kagi for privacy positioning, Tavily/Exa for agent-native results), documenting that this **shifts responsibility to the customer's chosen vendor**.
- **Default:** keep first-party DuckDuckGo routing as the balanced baseline.

6.5 Implementation note

Because web search is a **first-party** server, switching the upstream is a code/config change, not a runtime knob. **Decision (2026-06-10):**

- The **current web search server** is our specific solution (DuckDuckGo only) and stays as-is. We do **not** retrofit multi-provider behavior into it.
- A **separate multi-provider web search server** will be the **multi-provider** path (pluggable backends: DuckDuckGo / SearXNG / curated corpus / a third-party API), selectable per deployment. This keeps the current server simple and isolates provider complexity in the new one.

Until a multi-provider web search server exists, alternative providers are a build item, not a configuration toggle.

> **Note (research pass done 2026-06-10):** vendor privacy/retention/pricing were verified where possible in §6.6; remaining gaps are flagged *unverified* there and in chapter 9. Treat the "Why consider it" column above as positioning to confirm per deployment, not guarantees.

6.6 Verified provider details (research pass 2026-06-10)

Vendor-stated, list pricing as of 2026-06-10; re-verify at procurement. Items marked *unverified* could not be confirmed from an official page in this pass.

Provider	Index	Stated privacy / data control	Rough price	Airgap
SearXNG [SEARXNG]	none (proxies upstream)	no tracking/profiling; strips cookies; egress IP = your instance	free OSS + hosting	partial (needs egress to engines)

Provider	Index	Stated privacy / data control	Rough price	Airgap
Brave [BRAVE]	independe (own crawler)	SOC 2 Type II; ZDR offered (Enterprise); storing results needs a storage-rights plan	~$5 / 1k req	none
Kagi [KAGI]	premium/c	queries logged temporarily then auto-purged (logs 7d; Sentry 90d sampled); no public DPA/BAA or explicit "no-train on API" clause found *(partly unverified)*	~$12 / 1k req	none
Bing -> Grounding w/ Bing [BING-GROUNDING]	Microsoft	**data leaves the Azure compliance boundary; the MS Data Protection Addendum does NOT apply**; only query+params+key sent; legacy Bing API retirement ~2025-08-11 *(unverified)*; Grounding (classic) agents retire 2027-03-31	via Azure Foundry *(price unverified)*	none
Tavily [TAVILY]	agent-native	officially **SOC 2 certified, zero data retention** (FAQ); SOC 2 type + DPA not stated publicly *(gated)*	~$0.005-0.008 / credit	none
You.com [YOUCOM]	agent-native	SOC 2; "no model training"; ZDR (auto-purge); DPA-ready (vendor-stated)	~$5 / 1k req	none
Exa [EXA]	neural/emb	**default tier trains on query data**; **ZDR on Enterprise** (+ HIPAA); SOC 2 Type II; DPA via Trust Center	~$7 / 1k req	none
Perplexity (Sonar) [PERPLEXITY]	answers + citations	**ZDR by default** (no retention of Sonar API data, billing metadata only); **no training by default**; SOC 2 Type II + 2025 HIPAA gap assessment	tokens + ~$5-12 / 1k req	none
On-prem / curated corpus	your own	fully internal; no external egress	build/host cost	**yes**

Key research conclusions:

- **Only a curated/local index (internal RAG) is truly airgappable.** SearXNG is self-hosted but **not** airgapped - it has no index and must reach upstream engines. All seven cloud APIs require egress.
- **Best default data posture: Perplexity Sonar** (ZDR + no-training **by default**), **You.com** (ZDR, no-training, DPA-ready), and **Tavily** (officially SOC 2 + zero data retention). **Brave** and **Exa** offer ZDR **on Enterprise** (Brave needs a storage-rights plan to store results; Exa's **default tier trains on query data**). **Kagi** purges quickly but has no public DPA / no-train clause.
- **Compliance red flag - Bing Grounding:** Microsoft states data sent to Grounding with Bing **leaves the Azure compliance boundary and the Data Protection Addendum does not apply** - a poor fit for regulated customers despite the Azure brand.
- **Takeaway:** if a third-party API is unavoidable for a sensitive customer, prefer **default ZDR + no-training** providers and a signed DPA; otherwise prefer **SearXNG / on-prem corpus** (`chapter 6.4`).

6.7 Data-processing responsibility (GDPR-style)

Adopting a third-party search API does **not** offload accountability:

- The **adopting organization is the data controller** for whatever is in the query (and for the lawful basis of sending it).

- The **search vendor is a processor/sub-processor** (and often an independent controller for its own abuse/operational logs).

If a customer chooses a third-party provider, the policy should **require**: a signed **DPA**, **Zero Data Retention** where offered, a **no-training** guarantee, **region/residency** selection, and forwarding **minimal** query content (no PII/secrets). This is the concrete meaning of "responsibility shifts to the customer's chosen vendor" used elsewhere in this document.

> See sources and the full "unverified" list in `chapter 09`.

07 — Customer Responsibility & Policy

7.1 Shared-responsibility model

Web search safety is a shared responsibility. basebox provides the architecture, controls, and an honest risk picture; the customer chooses the risk posture and owns the residual risk that only they can control.

Responsibility	basebox	Customer
First-party routing, text-only ingestion, auditing	Yes	-
Untrusted-content framing, allow-list, scope reduction (as delivered)	Yes	-
Choosing **which apps** enable web search	Guidance	**Decision**
Keeping web search the **sole active tool** in those apps	Enforcement option	**Decision**
Deciding **who** (which group) may use it	Mechanism	**Decision**
Usage policy (e.g. no sensitive data in queries)	Template	**Owns + enforces**
Choosing a **third-party provider** (if any)	Options (`chapter 06`)	**Decision + vendor risk**
Accepting the **residual risk** of the open web	Disclosure	**Accepts**

7.2 Supported configurations

Tier	Configuration	Suitable for	Residual risk
A (recommended)	Web search as the **sole active tool** in a dedicated app + defined pilot group + usage guideline	Most customers, including regulated	Query leakage only (no exfiltration path)
B (advanced)	Web search co-enabled with other tools, **with** scope reduction + framing + approval	Customers who accept higher risk and have the controls delivered	Injection + bounded exfiltration
C (max privacy)	SearXNG self-hosted or curated on-prem corpus	Airgap / strict-privacy customers	Minimal-to-none open-web exposure
Unsupporte	Web search co-enabled with internal-data tools **without** scope reduction	-	Active exfiltration path; not offered

The decisive control is **Tier A's "sole active tool"** rule: with no second tool to read internal data, the combined-tools exfiltration path (risk P1b) does not exist, *regardless of model behavior*.

Compliance note (GDPR / HIPAA)

For regulated customers, the supported configurations map directly onto the compliance framing in `chapter 04.6`: keep web search **isolated (Tier A)**, enforce **data minimization** via the usage guideline (no personal data / PHI in queries), record the **controller/processor** split and any **DPA** for a chosen provider (`chapter 06.7`), and for the strictest cases prefer **Tier C** (on-prem corpus / self-hosted) so regulated data never leaves the environment. This is framing, not legal advice - confirm with counsel/DPO.

7.3 Recommended rollout

1. **Enable in dedicated research apps only**, where web search is the sole active tool.
2. **Limit access** to a defined pilot group.
3. **Publish a short usage guideline** (template below) - e.g. no patient or otherwise sensitive data in queries.
4. **Review** after a defined period (e.g. a few weeks): audit volume, incidents, feedback; then decide whether to widen.

7.4 Usage-guideline template (customer-facing)

> *Web search sends your query to an external search engine. Do not type patient names, case identifiers, or any confidential/personal information into a web search. Use web search for general, non-sensitive questions only. If you are unsure, do not include the detail.*

Customers should adapt this to their regulatory context (e.g. GDPR/HIPAA) and distribute it to the pilot group.

7.5 Pre-enablement checklist

- ☐ Web search is enabled only in apps where it is the sole active tool (Tier A), **or** Tier B controls are delivered and accepted, **or** a Tier C provider is in use.
- ☐ Access is limited to a defined group.
- ☐ A written usage guideline has been published to that group.
- ☐ The customer has been shown this document and has **acknowledged the residual risk**.
- ☐ A review date is set.
- ☐ (If applicable) the third-party provider choice and its data terms are recorded.

7.6 Note on the “can’t they just not use it?” reaction

A risk list alone implies the answer is “don’t enable web search.” That is not the policy. The policy is: **enable it in a supported configuration, with the customer’s informed consent to the residual risk.** Tier A makes that a reasonable, defensible default rather than an all-or-nothing choice.

08 — Conclusions

8.1 Findings

What the analysis (chapters 02-07) established:

- **F1 - The architecture is already privacy-sound.** Search runs through the first-party web search server to DuckDuckGo only; queries route through basebox on-prem (not the user's device); only plain text reaches the model (scripts/active content stripped); the user's IP is not exposed; no credentials are forwarded; every call is audited. The open issues are about *content trust* and *scope*, not the transport.
- **F2 - The dominant risk is exfiltration via combined tools (P1b).** Web search plus an internal-data tool in the same app lets an injected instruction read internal data and smuggle it out through a follow-up query.
- **F3 - The most realistic everyday risk is query leakage (P2).** Users may type sensitive data into queries; the query then leaves the premises. Free-text filtering is unreliable; scope + policy are the effective levers.
- **F4 - Prompt injection cannot be fully solved.** It is mitigated by defense-in-depth + scope reduction and is partly a property of the model (frontier models resist it better than smaller self-hosted models).
- **F5 - Active-content execution is already a non-issue (P5).** Text-only extraction removes scripts and active content.
- **F6 - The risk is the open web, not "search."** The DuckDuckGo-vs-DokuWiki comparison shows the confidentiality gap comes from egress to the open web; injection risk exists for any tool that brings untrusted text into context.
- **F7 - Some alternatives shift, rather than remove, risk.** A customer-chosen third-party provider moves responsibility to that vendor; a curated on-prem corpus removes open-web injection and leakage entirely.

8.2 Summary

Web search is the most-requested and least-containable MCP connector. Once attacker-influenceable content from the open web can enter the model's context, the system's safety cannot be *guaranteed*. The honest position is therefore not "it is safe," and not "do not use it," but: **we make web search as safe as it reasonably can be, we are precise about the limits, transparent about every risk we know or suspect, and clear about what we do and could do about each one - and the customer makes an informed, conscious decision and co-owns the residual risk.**

Concretely: routing is first-party and single-provider; ingestion is text-only; calls are audited; tools are gated. The planned controls (untrusted-content framing, host allow-list enforcement, tool-scope reduction, per-group visibility, human-in-the-loop) close the remaining gaps in priority order. Where the open web is not strictly required, a curated on-prem corpus removes the risk entirely.

8.3 Conclusions

- **C1 - Web search can be offered responsibly**, in a supported configuration, with informed customer consent - it does not have to be all-or-nothing.
- **C2 - The single most effective control is isolation:** running web search as the **sole active tool** in an app eliminates the exfiltration path (P1b) regardless of model behavior. This is the recommended default (Tier A, `chapter 07`).

- **C3 - The two cheapest, highest-value engineering wins** are untrusted-content framing and enforced host allow-listing; these should ship first.
- **C4 - "Best-effort" must never be sold as a guarantee.** We will not ship content/keyword filtering as a security control because it creates false confidence.
- **C5 - Model choice is part of the security posture.** With a smaller self-hosted model, the architectural and prompt-level controls matter more.
- **C6 - One written source of truth** (this single, universal document, used for all customers) keeps every customer conversation grounded in the same basis instead of re-opening it from scratch. For a given conversation, point the reader to the most relevant chapters (e.g. 01, 03, 07, 08) rather than creating a tailored variant.

8.4 Suggestions to our customers

Our goal is to give you everything you need to make a fully informed, conscious decision. With this document in hand, you can choose any of the following paths - and we are glad to help with each.

1. **Start now, in a controlled pilot.** Enable web search in a dedicated app where it is the **sole active tool**, limited to a defined **pilot group**, with a short **usage guideline** (no sensitive data in queries) and a **review date**. This is our recommended baseline (Tier A, `chapter 07`) and is suitable even for regulated environments.
2. **Talk to us about the roadmap.** If there is a specific safeguard you would like in place first (e.g. untrusted-content framing, host allow-listing, or stronger isolation between web search and internal-data tools), tell us - we can sequence it and tell you honestly where it stands.
3. **Bring your own requirements or ideas.** If you need maximum control, we can discuss a **self-hosted or customer-chosen search backend** (you gain control and take on the corresponding responsibility) or a **curated internal source** where the open web is not strictly required. We are happy to research options together (`chapter 06`).

What we ask in return

- Decide **which apps** may use web search and keep it the **sole active tool** there (or accept the documented higher-risk configuration).
- Decide **who** may use it, and publish the **usage guideline** to them.
- **Acknowledge the residual risk**: the open web means it can be made small but never zero (`chapter 07` checklist).

A note on timing

The topic of privacy is especially current. We would rather give you a clear, written, honest basis than a quick verbal assurance - this document is exactly that, and it is the same basis we provide to every customer.

8.5 Bottom line

Web search is the hardest connector to make safe, so it deserves the clearest policy. With first-party routing, text-only ingestion, the planned framing and scope controls, and a Tier A default, basebox can offer web search responsibly - honestly bounding, disclosing, and co-owning a residual risk that can be made small but never zero.

8.6 Decision record

A decision document should state the decision explicitly. **TODO:** fill owner / reviewers / date at sign-off.

Field	Value
Decision	Offer web search via the first-party DuckDuckGo path, defaulting to **Tier A** (sole active tool + pilot group + usage guideline), with explicit residual-risk acknowledgment.
Status	Proposed (pending review)
Owner / Decider	TODO
Reviewers	Jaro
Date	TODO
Supersedes	Ad-hoc analysis in YouTrack DEV-1115

Options considered

Option	Summary	Why not the default
0. Do not offer web search	Eliminates open-web risk entirely	Every customer wants it; refusing pushes users to shadow/unsafe tools and loses a key capability
1. Offer freely (any app, any tools)	Maximum utility	Unsupported - leaves the exfiltration path (P1b) wide open
2. Tier A default (chosen)	Web search as sole active tool + pilot + guideline	Recommended balance: removes P1b, bounds P2
3. Tier C only (on-prem corpus / SearXNG)	Maximum privacy	Best for strict cases, but denies open-web search to those who need it; higher ops cost

"Do nothing" (Option 0) is included deliberately as the baseline against which the recommendation is judged.

Pilot success / exit criteria

A pilot (Tier A) is considered successful enough to widen if, over the review period:

- **Zero confirmed** exfiltration or regulated-data-leakage incidents.
- Any prompt-injection occurrences are limited to **misleading answers** (no data loss) and are caught in review.
- Audit volume and user feedback are within expectations; usage guideline is being followed.
- No unresolved High residual risk for the customer's context.

If these are not met, **do not widen**; remediate (add controls, tighten scope) or fall back to Tier C.

What we will NOT claim (guardrail for customer conversations)

- We will **not** claim web search is "safe" or that prompt injection is "solved"/"prevented."
- We will **not** present content/keyword filtering as a security guarantee.
- We will **not** imply that choosing a third-party provider removes the customer's data-controller responsibility.
- We will **not** promise unshipped controls as if they are in place; roadmap items are labeled as such (`chapter 05`).

09 — References

9.1 External references

Standards, frameworks, vendor documentation, and research consulted for this document. Accessed 2026-06-10 unless noted.

Standards & frameworks

- **[OWASP-LLM]** OWASP. "OWASP Top 10 for LLM Applications 2025." OWASP GenAI Security Project, 2025. https://genai.owasp.org/llm-top-10/ — technical taxonomy used in ch. 04.5 (LLM01 Prompt Injection → P1/P1b; LLM02 Sensitive Information Disclosure → P2; LLM05 Improper Output Handling → P5; LLM06 Excessive Agency → scope/P1b; LLM09 Misinformation → manipulated answers).
- **[NIST-RMF]** NIST. "AI Risk Management Framework (AI RMF 1.0)." NIST, 2023. https://www.nist.gov/itl/ai-risk-management-framework — plus **NIST-AI-600-1 Generative AI Profile** (2024-07-26) and the Critical-Infrastructure profile concept note (2026-04-07). Governance backbone (Govern/Map/Measure/Manage) in ch. 02.7.

Research

- **[QI-2026]** Qi et al. "Towards Trustworthy Agentic AI." Academia AI and Applications, 2026-04-29 (CUHK, Fudan, Shanghai Academy of AI for Science). https://www.academia.edu/3071-0286/2/2/10.20935/AcadAI8260 — peer-reviewed survey formalizing the lethal trifecta, a five-stage agent lifecycle, and six non-negotiable controls; argues model-layer defenses cannot stop data-layer attacks (data-layer enforcement required). Anchors ch. 04 (P1b) and ch. 05.
- **[GRESHAKE-2023]** Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, Mario Fritz. "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection." Proc. 16th ACM Workshop on AI and Security (AISec '23), 2023. arXiv:2302.12173. https://arxiv.org/abs/2302.12173 — peer-reviewed foundation for indirect prompt injection; concludes no single defense fully prevents it, recommending defense-in-depth. Anchors ch. 04 (P1/P1b) and ch. 05.
- **[LIU-2024]** Y. Liu et al. "Formalizing and Benchmarking Prompt Injection Attacks and Defenses." USENIX Security, 2024. arXiv:2310.12815. https://arxiv.org/abs/2310.12815 — first formal framework + unified benchmark; no evaluated defense fully prevents injection.
- **[ZHAN-2024]** Q. Zhan et al. "InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated LLM Agents." Findings of ACL, 2024. arXiv:2403.02691. https://arxiv.org/abs/2403.02691 — benchmark splitting attacker intent into direct harm vs private-data exfiltration; peer-reviewed anchor for ch. 04 P1b.
- **[DEBENEDETTI-2024]** E. Debenedetti et al. "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents." NeurIPS Datasets & Benchmarks, 2024. arXiv:2406.13352. https://arxiv.org/abs/2406.13352 — dynamic agent-security benchmark; motivates robust-by-design.
- **[CAMEL-2025]** E. Debenedetti et al. "Defeating Prompt Injections by Design" (CaMeL). arXiv:2503.18813, 2025. https://arxiv.org/abs/2503.18813 — control/data-flow separation + capabilities for provable injection resistance independent of the model. Supports ch. 05 (tool-scope reduction).
- **[BEURER-2025]** L. Beurer-Kellner et al. "Design Patterns for Securing LLM Agents against Prompt Injections." arXiv:2506.08837, 2025. https://arxiv.org/abs/2506.08837 — provably-resistant design patterns; structure over prompt-level guardrails.

- **[HINES-2024]** K. Hines et al. (Microsoft). "Defending Against Indirect Prompt Injection Attacks With Spotlighting." arXiv:2403.14720, 2024. https://arxiv.org/abs/2403.14720 — data-instruction separation (delimiting/datamarking/encoding); attack success >50% to <2%. Supports ch. 05 (untrusted-content framing).
- **[HOU-2025]** X. Hou et al. "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions." arXiv:2503.23278, 2025. https://arxiv.org/abs/2503.23278 — canonical MCP-security systematization (lifecycle + threat taxonomy). Supports ch. 02.7.
- **[SONG-2025]** "Beyond the Protocol: Unveiling Attack Vectors in the Model Context Protocol (MCP) Ecosystem." arXiv:2506.02040, 2025. https://arxiv.org/abs/2506.02040 — first end-to-end empirical study of MCP attacks.
- **[MA-2025]** S. Ma et al. "Safety at Scale: A Comprehensive Survey of Large Model and Agent Safety." arXiv:2502.05206, 2025. https://arxiv.org/abs/2502.05206 — broad (706-paper) safety taxonomy; backstop survey.

Industry & practitioner

- **[WILLISON-2025]** S. Willison. "The lethal trifecta for AI agents: private data, untrusted content, and external communication." simonwillison.net, 2025-06-16. https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/ — authoritative practitioner source that coined the "lethal trifecta" (and "prompt injection").
- **[KITEWORKS-2026]** P. Spencer. "AI Agent Security: The Lethal Trifecta Explained." Kiteworks, 2026-06-01. https://www.kiteworks.com/cybersecurity-risk-management/ai-agent-security-lethal-trifecta/ — explainer of [QI-2026]; cites EchoLeak (CVE-2025-32711, M365 Copilot zero-click exfiltration) and agent-skill vulnerability statistics.

Model providers (safety & data posture)

- **[ANTHROPIC-TRANS]** Anthropic. "Anthropic's Transparency Hub." Anthropic, 2026. https://www.anthropic.com/transparency — model reports with prompt-injection, defense-in-depth, and data-handling statements (quoted in 9.2).
- **[ANTHROPIC-FABLE]** Anthropic. "Claude Fable 5." Anthropic, 2026-06-09. https://www.anthropic.com — current frontier model; 30-day safety retention; US-only inference option; accompanying system card. Cloud deployments use Claude Opus 4.8 (ch. 05.6).
- **[OPENAI-GPT4O]** OpenAI. "GPT-4o System Card." OpenAI, 2024-08-08. https://openai.com/index/gpt-4o-system-card/ — Preparedness Framework (deploy only if ≤ Medium); moderation classifiers over inputs and outputs; external red teaming. Used in ch. 05.6.1.
- **[GOOGLE-SAIF]** Google. "Google's Secure AI Framework (SAIF)." Google Safety Center, 2026. https://safety.google/cybersecurity-advancements/saif/ — six core elements; risk self-assessment; Google AI Red Team; CoSAI coalition. Used in ch. 05.6.1.
- **[XAI-PRIV]** xAI. "xAI Privacy Policy." X.AI LLC, effective 2026-04-04. https://x.ai/legal/privacy-policy — asks users not to submit personal data; consumer vs API/business split (BAA/DPA); Grok uses internet search + public X posts; 30-day Private-Chat deletion. Used in ch. 05.6.1.

Search providers (privacy, retention, pricing) — see ch. 06.6

- **[DDG-PRIV]** DuckDuckGo. "Privacy Policy." Duck Duck Go, Inc., updated 2026-06-03. https://duckduckgo.com/privacy — no tracking/profiling; IP not logged to disk tied to searches; anonymous queries only; does not sell personal data. Supports P3 (ch. 03/04).
- **[BRAVE]** Brave. "Brave Search API." https://brave.com/search/api/ — independent index; SOC 2 Type II; Zero Data Retention on Enterprise; storing results requires a storage-rights

plan.

- **[PERPLEXITY]** Perplexity. “Privacy & Security.” https://docs.perplexity.ai/docs/resources/privacy-security — Sonar API: Zero Data Retention and no model training by default; SOC 2 Type II; 2025 HIPAA gap assessment.
- **[YOUCOM]** You.com. “API Pricing / Security.” https://documentation.you.com/pricing — SOC 2; no model training; Zero Data Retention (auto-purge); DPA-ready.
- **[TAVILY]** Tavily. https://docs.tavily.com/ — officially SOC 2 certified with zero data retention; SOC 2 type and DPA not publicly enumerated.
- **[EXA]** Exa. “Privacy Policy” / “Security.” https://exa.ai/privacy-policy ; https://docs.exa.ai/reference/sec — SOC 2 Type II; DPA via Trust Center; default tier uses query data for product improvement / model training; Zero Data Retention is an Enterprise feature (+ HIPAA option).
- **[KAGI]** Kagi. “Privacy.” https://kagi.com/privacy — queries logged temporarily then auto-purged; no public DPA/BAA or explicit “no-train on API” clause found.
- **[BING-GROUNDING]** Microsoft. “Grounding with Bing Search” (Azure AI Foundry). https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/bing-grounding — data sent to Grounding with Bing **leaves the Azure compliance boundary and the Microsoft Data Protection Addendum does not apply**; Grounding (classic) agents retire 2027-03-31. The standalone Bing Search APIs are being retired in favour of this service (exact date not confirmed here).
- **[SEARXNG]** SearXNG documentation. https://docs.searxng.org/ — self-hosted metasearch; no built-in index (proxies upstream engines); no tracking/profiling.

9.2 Key cited statements

Direct quotes from Anthropic-published model reports at anthropic.com/transparency, used in chapters 04 and 05 to ground the prompt-injection and defense-in-depth points:

- **Prompt injection, defined (Claude 4):** “attacks where malicious content on websites or in documents tries to trick Claude into doing things the user never asked for — like copying passwords or personal information.”
- **Defense-in-depth + red-team result (Claude 4):** “Claude Opus 4 achieving an 89% attack prevention score (compared to 71% without safeguards) and Claude Sonnet 4 achieving 86% (compared to 69%).”
- **Untrusted-content framing (Claude 3.7 Sonnet):** “websites or documents might contain hidden text that tries to trick Claude into doing things the user didn’t ask for, called ‘prompt injection’.”
- **Per-surface robustness / tool scope (Claude Sonnet 4.5):** with mitigations, MCP 94% / computer-use 82.6% / bash tool-use 99.4% of attacks prevented; lowest injection rate among 23 models in an external red team.
- **Browser red-team (Claude Opus 4.5):** Chrome extension, adaptive attacker with 100 attempts; only 1.4% successful vs 10.8% for Sonnet 4.5 with previous safeguards.
- **Residual risk is never zero (Claude 3.7 Sonnet):** “our safety systems block 88% of these attempts, compared to 74% with no safety systems … We aim to continue enhancing our safety systems.”
- **No training on user data (Claude 3.7 Sonnet):** “We did not train this model on any user prompt or output data submitted to us by users or customers.”
- **Over-eager autonomy / human-in-the-loop (Claude Opus 4.6):** model “at times too eager, taking risky actions without asking first” (e.g. sending emails, using auth tokens); “We have made changes to Claude Code to help address this.”

9.3 Internal sources

YouTrack articles & issues (basebox internal):

- **[BB-INS-IR]** YouTrack **INS-A-76** — "Incident Response Process" (basebox ISO 27001 process). Referenced by ch. 05.7. Related: **INS-A-1** "ISO 27001 Controls"; **INS-A-79 / INS-A-80** Risk Analysis / Treatment (internal risk-assessment method and data-exfiltration entries).
- **[BB-MCP-SEC]** YouTrack **DEV-A-16** — "MCP definition and resources": internal MCP security notes (OAuth/identity, local access control, network isolation, data-boundary enforcement, prompt-injection & data-leak tooling). **DEV-A-18** — "MCP Server — best practices" (MCP-as-agent-UI, outcome-oriented tools, security guides). **DEV-A-156** — "The basebox MCP Manifest — Concept, Catalog, and Delivery."
- **[BB-MODELS]** YouTrack model/inference notes (context for ch. 05.6): **DEV-A-83** "LLM Selection and Inference Engines," **DEV-A-75** "On-premise Model Switching and Serving Multiple Models," **DEV-A-14** "LLM Recommendations for On-Premises Deployment"; plus the three-tier model-config work (Felix Raab; DEV-312/335/336/351 …) and the Audit Log API guide (DEV-691).
- **DEV-1115** — "Create risk assessment table for MCP" and its comment thread (mitigation tables, customer-facing draft, DuckDuckGo-vs-DokuWiki comparison) — the internal analysis this document consolidates.

Source code (internal reference, verified 2026-06-10):

- **Web search server** (MCP catalog): text extraction module (noise tag stripping, block tag boundaries); the shared MCP credential framework; the MCP manifest spec.
- **AI service** (platform backend): MCP audit event implementation (`mcp.tool.invoke`); tool settings schema (per-tool, per-user, per-app, including `is_enabled` and `requires_human_approval`); baked-in MCP catalog directory.

9.4 Citation format

Inline references use the bracketed keys above (e.g. `[OWASP-LLM]`). Full entries list: author, title, publisher/site, year, URL (accessed 2026-06-10).