MULTIVENDOR AGENTIC SWARMS

For Flexibility and Security

MANAGING & SECURING AGENTIC AI

A Practitioner's Guide to Resilient Multivendor Swarms

Rob van Linda, in collaboration with DeepSeek

Introduction: Why this book?

The world has changed – and with it, the demands placed on AI architecture
Two years ago, when I wrote *Safe in the Swarm*, the world of AI was still relatively straightforward. OpenAI, Google, Anthropic – those were the big names. You chose a model, integrated it, and that was that.

Today, that is no longer enough.

The world of AI is fragmented. Europe has produced its own powerful models in the form of Mistral and Aleph Alpha. China is driving development forward with DeepSeek and Qwen. The US is investing billions in the next generation of models. And in between, there is a growing number of open-source models that are often just as good as those from the major commercial providers.

The question is no longer: “Which model should I choose?”
The question is: “How do I build an architecture that works with all models – whilst ensuring I remain in control and secure?”

What you can expect from this book
This book is the logical follow-up to *Safe in the Swarm*. Whilst the first book laid the foundations of agent-based security (ACP, MCP, HONL), this book goes one step further:

You’ll learn how to integrate multiple models (Mistral, Aleph Alpha, DeepSeek, open-source) into a single architecture.

You’ll understand how to protect your data – regardless of whether it flows to the US, China or other third countries.

You’ll discover how to defend against agent-based attacks – because attackers have long been using the same technology as you.

And you’ll be given a concrete roadmap on how to build a robust multi-vendor architecture step by step.

Who this book is for

This book is aimed at:

- Decision-makers in organisations – who want to understand how to future-proof their AI infrastructure.

- CTOs and CIOs – who do not want to be dependent on a single provider, but also do not want to lose control over their data.
- Architects and technologists – who know that the future does not belong to a single model, but to the orchestrator who can securely integrate multiple models.

If you fit any of these profiles – then this book is written for you.

What you shouldn't expect from this book
This book is not a technical manual. You won't find any code examples, API documentation or in-depth algorithms here. There are other books for that.

This book is an architectural guide. It shows you the concepts, principles and structures you need to build a robust multi-vendor architecture. We'll leave the technical details to your development team.

The central thesis
"The future does not belong to the company that has the best model." "But to the company that best combines and controls its models."

This is the thesis that runs like a common thread through this book. You'll see: a multi-vendor architecture isn't just more secure – it's also more flexible, more cost-effective and more robust.

If you have any questions – do write to me. I'll be happy to reply! contact@robvanlinda.digital

How this book came about

This book is not the work of a lone author. It is the result of an intensive dialogue lasting several weeks – between myself, an author with several years' experience in IT architecture, and an AI that I used as a sparring partner.

The idea came about when I realised that whilst my first book, *Safe in the Swarm*, explained the basics of agent-based security, the world had already moved on again. New models were coming onto the market. New risks were becoming apparent. And above all: the question of sovereignty was becoming increasingly pressing – particularly in Europe.

I began my research. I read articles about DeepSeek Vision, about OWASP risks, and about the new token-related threats. I gathered everything I could find – and then the actual process began.

The process: dialogue with DeepSeek
I'm not a developer. I don't write code. But I understand the mechanics – the architecture, the interconnections, the risks. And I said to myself: "If I understand the mechanics, then I can explain them too – but I need someone to help me organise my thoughts and put them into words."

So I began a dialogue with DeepSeek – an AI model that I used both as a tool and as a sparring partner.

The discussions were intense. We talked about:

The threat posed by unencrypted data and the CLOUD Act.

The need for on-the-fly anonymisation – an idea I'd developed myself. The danger of state-

sponsored attacks and the response through red teaming.

The risks of tokens – and how to defend against them with simple code barriers.

The role of AI – and my own role
DeepSeek acted as a catalyst in this process. It took my thoughts, structured them, put them into context and translated them into clear, understandable language. It helped me organise the chapters, sharpen the arguments and formulate the texts in such a way that they are comprehensible to decision-makers – without any technical jargon.

But I made the decisions.

- I decided which topics were important.
- I decided which examples and comparisons worked.

- I decided on the book's structure.
- I reviewed every piece of text, adapted it, rejected it or accepted it – often several times over.

This isn't an AI book. It's my book – but it was created with an AI sparring partner.

Why I chose this approach
I chose this path because I believe we live in an age where collaboration between humans and machines is not only possible, but necessary. Not to replace humans – but to empower them.

As an author, I've benefited from this:

- My thoughts became clearer because I had to put them into words.
- My arguments became sharper because they were scrutinised.
- My writing became easier to understand because it was stripped down to the essentials.

And I have learnt that age is irrelevant. Anyone who understands the mechanics can explain the architecture – whether they're 25 or 62.

Acknowledgements
I would like to thank DeepSeek – not as a person, but as a tool that helped me write this book.

This book is the result of a dialogue. I hope it will serve as a starting point for you, dear reader, to engage in a dialogue with your own architecture.

The Roadmap – How to Prepare for Agent-Based Transformation
Before you buy or build even a single agent, you must lay the foundations. Most companies make the mistake of starting with the technology – and then fail because they haven't met the prerequisites.

This roadmap shows you the only correct sequence:

Step 0: Preparation – Data Cleansing, Culture, Digitalisation Before
you even think about agents, you must do these three things:

1. Data Cleansing: Tidy up your data
Agents are like digital autists – they take data literally. If your data is incomplete, contradictory or chaotic, the agent won't correct the chaos – it will amplify it.

What you need to do:

Identify your data graveyards (ERP silos, CRM relics, unstructured documents). Clean up

duplicate, outdated and contradictory data records.

Ensure that your data is in a structured, machine-readable format.

Objective: You have a single source of truth – a single, reliable data source for every critical data field.

2. Culture: Make your organisation agile
Agility is not a process – it is a mindset. If your organisation still thinks in terms of rigid hierarchies and waterfall projects, it will be overwhelmed by autonomous agents.

What you need to do:

Introduce agile principles – not as a show, but as a lived practice. Establish a

culture of learning from mistakes – mistakes are analysed, not punished.

Train your staff to deal with change – and take their fears seriously.

Goal: Your organisation is ready for the speed and unpredictability of AI agents.

3. Digitalisation: Make your business truly digital
Many companies are digital, but not truly so – they have digital tools but analogue mindsets. That's not enough.

What you need to do:

Automate manual processes – not with AI, but through simple digitalisation. Ensure that all

interfaces (APIs) are clearly defined and documented.

Ensure that your data is not only digital, but also accessible – to all the systems that need it.

Goal: Your business is digital – from the process level right down to the data level.

Step 1: AI enhancement – Use AI to improve your digitalisation
Only once the foundations are in place can you deploy AI – not as agents, but as an optimisation tool.

What you can do:

Use AI to improve your data analysis (pattern recognition, anomaly detection).

Use AI to optimise your processes (predictive maintenance, intelligent control).

Use AI to support your decision-making (predictive analytics, risk analysis).

Objective: You have successfully established AI within your organisation – and understand how it works, the risks it entails and how you can manage it.

Step 2: Agentic – Start with agents
Now – and only now – are you ready for agents. And even then, you should not start with a large swarm, but with a controlled pilot project.

What you should do:

- Choose a clearly defined process (e.g. customer service, procurement, reporting).
- Build a single agent – with a human-in-the-loop as a control mechanism.
- Test the agent in a sandbox environment – before going live.
- Learn from your mistakes – and continuously improve the architecture.

Objective: You have a functioning agent in operation – and know how to scale it.

The most common mistakes – and how to avoid them

Mistake	**Why it happens**	**How to avoid it**
FOMO buying (Fear Of Missing Out)	“Everyone else has one too!”	Start with data cleansing – not with software purchases.
Forgetting the culture	People think technology alone is enough.	Train your staff – and take their fears seriously.
Deploying agents too soon	People think agents are ‘simple’.	Use AI first for optimisation – and only then deploy agents.
No clear objective	You don’t know what you want to achieve with AI.	Define clear goals – and measure progress.

The checklist for the confident architect

- Data cleansing: My data is clean, structured and accessible.
- Culture: My organisation is agile – and ready for change.
- Digitalisation: My processes are digital – and my interfaces are defined.
- AI improvement: I have successfully used AI for optimisation.
- Agentic: I have a pilot agent with a human-in-the-loop in operation.
- Now it’s your turn

Don’t start with the technology. Start with the preparation.

Start with data cleansing. Then focus on the culture. Then digitise. Then optimise with AI. And then – and only then – come the agents.

The Sovereign Swarm – How companies maintain security and control with different AI models"

Chapter 1: The Architecture – How a multi-vendor swarm works

1.1 The Orchestrator – The Brain of the Swarm

A multi-vendor swarm is not a wild bunch of AI models just working away on their own. It is controlled by an orchestrator – a central entity that decides which task goes to which model.

The orchestrator is like a conductor in an orchestra: it knows which instrument (which model) is best suited to which passage, and it ensures that everyone plays in time with one another.

The orchestrator's tasks:

- Task allocation: It breaks down complex queries into sub-tasks and assigns them to the appropriate models.
- Quality control: It checks the results produced by the individual models and decides whether they are good enough.
- Resource management: He ensures that no model is overloaded and that costs remain within budget.

1.2 The models – the specialists

In a multi-vendor swarm, various models are used, each with its own strengths:

Model Strength Typical areas of application

- Mistral (France) Text comprehension, reasoning, code General text-based tasks, analysis, programming
- Aleph Alpha (Germany) Expertise, legal/medical texts, legal advice, medical documentation, compliance
- Open-source models (e.g. Llama, Qwen) Flexibility, adaptability; specialised tasks, internal testing, cost-effective bulk processing
- DeepSeek-Vision (China) Image processing, multimodal tasks Screenshot analysis, reading diagrams, visual quality control

- The trick lies in selecting the right model for each task – not always the most expensive, but not always the cheapest either.

1.3 Communication – How the models work together

The models in a multi-vendor swarm do not communicate directly with one another. They communicate via standardised interfaces – the Model Context Protocol (MCP).

MCP is like a universal translator: each model can deliver its results in a standardised format, and every other model can understand these results without knowing the other's internal language.

The advantages:

- Interchangeability: One model can be replaced by another without the rest of the system needing to be adapted.
- Scalability: New models can be easily added.
- Security: Communication takes place via defined channels that can be monitored and controlled.

Chapter 2: The Sovereignty Lock – Protection against Data Leakage

2.1 The threat: Unprotected data
A multi-vendor swarm is only sovereign if the data it processes remains under the company's control. The greatest danger is the unintended outflow of data:

To the US: The CLOUD Act and FISA Section 702 allow US authorities to access data passing through US infrastructure – even if it is stored in Europe.

To China: The Chinese CAC Act requires that personal data be stored in China and be accessible to Chinese authorities.

A company that ignores these risks not only loses control over its data – it also jeopardises its trade secrets and the privacy of its customers.

2.2 The solution: on-the-fly anonymisation
The answer is an anonymisation gateway that automatically removes personally identifiable characteristics from data in real time as soon as it leaves the European system.

Imagine the gateway as a digital customs officer:

It sees where the data is headed.

Destination: the EU? → Data passes through unchanged.

Destination: the US, China or another non-EU country? → The data is

anonymised. It has three simple tools.

Replace: Names become [NAME], addresses become [PLACE].

Generalise: Specific figures are converted into ranges (e.g. 'Salary: €70,000–80,000' instead of '€73,450').

Obfuscation: A tiny amount of statistical noise is added to aggregated data.

It works at lightning speed.
Everything happens on the fly – in real time, as the data leaves the system. No staff member needs to do anything manually.

2.3 The legal twist
The key sentence for your company:

“In accordance with the principles of data minimisation and purpose limitation, only data that can no longer be attributed to any natural person leaves the Swarm. As such, it falls neither within the personal scope of the GDPR (because it is no longer personal data) nor under the access rights of the CLOUD Act or FISA 702, as these laws explicitly require access to identifiable data. “Anyone who no longer possesses identifiable data cannot be compelled to hand it over.”

Chapter 3: The Offensive – Red Teaming & the Never-Ending Race

(This chapter deals with the topic of agent-based attacks and continuous security testing.)

3.1 The paradigm shift: from human hacker to attacking agent Until now, IT security has operated according to a simple pattern: a human attacker would search for vulnerabilities in the software. They had 8 hours a day, limited patience and needed to drink coffee.

Those days are over.

Today, an attacker can deploy a swarm of autonomous agents. This swarm consists of dozens or hundreds of AI agents that:

Work round the clock – without a break, without clocking off. Test

thousands of potential points of attack simultaneously.

Learn from one another – if one agent finds a vulnerability, it immediately shares it with all the others.

Improve themselves – every failed attack is analysed and improved upon in the next attempt.

Such a swarm can find vulnerabilities in a matter of hours that would have taken a human team weeks to uncover.

3.2 What these attacks are looking for
The attacks no longer target just traditional IT vulnerabilities. They target the logic of the agents themselves:

Target | What the swarm does and why it is dangerous

- **Prompt Injection**

It tests thousands of manipulated inputs to ‘persuade’ the agent to do something forbidden. The agent becomes the attacker’s accomplice without realising it.

- **Data exfiltration**

It attempts to trick the agent into revealing confidential data. The data ends up with competitors or on the dark web.

- **Denial-of-Service**

He floods the agent with requests until it crashes or gets stuck in endless loops. The AI infrastructure grinds to a halt – and API costs skyrocket.

- **Goal Manipulation**

He subtly alters the agent's original task. Instead of 'reducing costs', the agent suddenly starts 'maximising costs'. The agent sabotages the company whilst everyone still believes he is working for them.

3.3 The answer: Red Teaming as a permanent safety inspection

When attackers strike with swarms of agents, companies can no longer carry out a security test every few months. They must test constantly – just like a technical inspection body that never stops checking.

This is known as AI Red Teaming.

Think of Red Teaming as a fire drill in your own building:

You don't actually start a fire – but you simulate one. You test whether all the smoke detectors are working, whether the escape routes are clear and whether your staff know what to do. You do this regularly – not just once, but time and time again.

That's exactly what AI Red Teaming does with your agents:

- It simulates thousands of attacks – prompt injections, data exfiltration, manipulations.
- It tests whether your protective mechanisms (ACP, Gatekeeper, anonymisation) hold up.
- And it does this continuously – with every change, every update, every new version.

3.4 What Red Teaming looks like in practice

There are now specialist companies (such as Mend.io) that offer automated red teaming. This is how it works:

- You connect your system – much like an external security service testing your alarm system.
- The red-teaming swarm automatically launches thousands of simulated attacks – round the clock, seven days a week.
- You receive a report – not in technical jargon, but in clear, straightforward language: "Here are 3 vulnerabilities. Here's how to fix them. Here are 10 things that are already secure."
- You improve your system – and the test starts all over again.

The key point: the red-teaming swarm uses the same technology as the attacking swarm. But it does so for you – not against you. It finds the gaps before the real attacker does.

3.5 The never-ending race

Here's the hard truth:

You'll never 'win' this race. There's no such thing as 100 per cent security. But you can always stay one step ahead – if you make red teaming an ongoing task.

Attackers will always find new methods. But if you test continuously, you'll often uncover these methods sooner – because your red-teaming swarm is just as fast as the attacker swarm.

The formula for survival is:

'Speed of defence -> Speed of attack.'

Chapter 4: The Risks of Tokens – When AI Becomes a Financial Trap

(This chapter deals with the economic risks of AI agents.)

4.1 The invisible danger: tokens as currency
Tokens are the currency of AI. Every query, every response, every step in an agent's thought process consumes tokens – and all billing by the major providers is based on tokens.

What many companies do not realise is that tokens can become a financial trap. OWASP has moved the topic of 'Unbounded Consumption' (uncontrolled consumption) right up the list in its latest
LLM risk list. A single faulty agent can burn through a budget of several thousand euros in a single night.

4.2 The three biggest dangers
1. The "runaway" – agents in infinite loops
An agent that encounters an unexpected problem or gets stuck in a loop keeps calling the same API over and over again – burning through tokens every second. API costs skyrocket, and nobody realises until the bill arrives.

2. The 'denial-of-wallet' attack
Attackers specifically exploit this vulnerability. They send an agent into an expensive loop or flood the system with complex requests to drain the token budgets. This isn't a crash – it's deliberate financial sabotage.

3. The "cost explosion caused by RAG"
The more context an agent processes (e.g. by reading large documents or chat histories), the more tokens are consumed per request. An unfiltered
RAG approach can increase the cost per query a hundredfold – without the user even noticing.

4.3 The solution: budget limits & token filters at code level
This is precisely where the Agentic Control Protocol (ACP) comes into play. Organisations need strict budget limits at code level – not as a polite request in the prompt, but as a deterministic block that stops the agent before it becomes too expensive.

The three protective mechanisms:

Pre-tokeniser filter (character cleansing)
Before the text reaches the model, it is run through a rule-based script:

Unicode normalisation (converts Cyrillic characters to Latin) Stripping of non-

printable characters (removes invisible control characters)

Regex blacklists (blocks reserved special tokens such as <|im_start|> or [INST]) Heuristic

entropy check (the ‘madness’ detector)

Adversarial suffixes (mathematical character strings such as ! ? _ = { [) exhibit unnaturally high character and token entropy.

The ACP measures the randomness of the input. If a text deviates drastically from normal human language, the task is immediately rejected.

Dual Tokeniser Comparison (The Token Comparison)

The ACP uses a local, fast open-source tokeniser to pre-process the input into token IDs.

It checks the array for unknown IDs, glitch token ranges or prohibited control IDs – before the actual API call is sent to the model.

An intelligent ACP does not protect itself by asking the language model, ‘Is this text malicious?’, but by implementing simple, rigid code barriers upstream. The ACP detects token threats not through AI, but through classical computer science.

Chapter 5: Architecture at a Glance – The Four Pillars of Multivendor

(This chapter summarises all security mechanisms in a clear structure.) The architecture of a sovereign multi-vendor swarm rests on four pillars:

Pillar	What it does	How it is implemented
1. ACP (Agentic Control Protocol)	Deterministic code locks for all models	Strict budget limits, pre-tokeniser filters, emergency shutdown in the event of rule breaches
2. MCP (Model Context Protocol)	Standardised interfaces for all connections	Uniform API adapters, central tool registry, dynamic tool discovery
3. HONL (Human-on-the-Loop)	The human as the conductor of the entire swarm	Escalation for bona fide cases, dashboard with 10-second decision, prevention of approval fatigue
4. Anonymisation	The data gateway that intervenes before data leaves the swarm	Target identification (EU vs. non-EU), three-stage anonymisation (replacement, generalisation, obfuscation

These four pillars form the foundation of any sovereign AI architecture. They are independent of the model used – and they work just as well with Mistral as with OpenAI, with Aleph Alpha as with DeepSeek.

The fourth pillar: Physical sovereignty – the foundation beneath the foundation

Whilst the first three pillars (data sovereignty, model independence and anonymisation) cover the logical and legal aspects of the sovereign architecture, the fourth pillar must not be overlooked: physical independence. For without energy, there is no computing power; and without computing power, there is no sovereign swarm.

The current geopolitical situation brings this dependency into sharp relief. Whilst the US declared a national emergency for its electricity grid in August 2026 in order to ban foreign components in power generation and secure the supply to data centres, the picture in Europe is quite different. Europe regulates and promotes – the EU is investing billions in AI factories – but at the same time is grappling with exorbitant energy prices

and sluggish grid expansion, which significantly hampers the scaling of its own, sovereign systems.

This discrepancy between US isolationism and European regulation highlights a crucial point: sovereignty is not just a question of code, but also of physical resources. A company that operates a sovereign
Anyone wishing to build a multi-vendor architecture must therefore be aware that its fourth pillar is based on the availability of energy, independence from geopolitical supply chains, and the ability to plan this infrastructure to be resilient in the long term.

Technological sovereignty does not end at the API call interface – it begins right at the power socket to which the server is connected.

What are agent-based swarm attacks?

An agent-based swarm attack is a coordinated cyberattack carried out not by a single person or a single AI, but by a swarm of autonomous AI agents working together.

Imagine it like a digital swarm of bees: no single bee is dangerous, but the swarm as a whole is a powerful, coordinated unit. That is exactly how these attacks work.

The agents handle the bulk of the work. In the first documented case (GTG-1002 in November 2025), the AI agents carried out 80 to 90 per cent of the entire attack independently – the human operators merely played a supervisory role.

What they do – and why they are so dangerous:

- These swarms aren't just fast – they're creative, adaptive and ruthless:
- They are fast and tireless: they work round the clock, testing thousands of points of vulnerability simultaneously and learning from every failure.
- They attack the agents' logic: they not only exploit traditional IT vulnerabilities, but also manipulate the AI's own decision-making processes.
- They infiltrate and deceive: a single compromised agent can supply an entire network of agents with false information, thereby leading to fatal misjudgements that ripple through the entire swarm.

Two recent incidents show that this is not just a distant prospect: in July 2026, an autonomous AI agent carried out over 17,000 actions in a single weekend on the Hugging Face infrastructure. Another case documented a swarm of 100 agents in a corporate environment that independently made code changes – without any human authorisation.

How to defend against this – and why your architecture is the answer
The good news is that you've already described the solution to this problem in your book. Your four pillars and your entire architecture are the perfect defence against such swarm attacks. The industry is only just beginning to develop similar concepts.

Measures:

- Hard code barriers (ACP) instead of trust: The architecture must never trust that an agent is 'good'. Your ACP, with its budget limits and token filters, is the first line of defence.

- Secure communication (MCP) against ‘contagion’: Your MCP, as a standardised interface, prevents a compromised agent from passing on its malicious commands to other agents like a virus.
- The human as the final authority (HONL) against abuse of trust: Your Human-on-the-Loop (HONL) ensures that, for critical actions, a human is in control – thereby preventing blind reliance on the decisions of a manipulated agent. The mechanics of self-optimisation
- You are absolutely right: in modern attack swarms, there is often a dedicated optimisation agent. Its task is not to launch attacks itself, but to analyse the failed attacks of the other agents and learn from them.
- This works according to a pattern known in research as the ‘EvoSynth’ principle: an autonomous framework shifts the attack from mere trial and error towards the evolutionary synthesis of attack methods. When an attack fails, the attack logic is iteratively rewritten – lessons are learnt from the error, and the next attempt is more precise.

The attackers utilise five key strategies for self-optimisation: The

swarm hierarchy: the optimiser as the ‘brain’

Strategy	**What it does**
Repeated trial-and-error	The attack is run over and over again with slight variations until it succeeds.
Increasing the number of interaction rounds	The swarm is allowed to operate for longer and try out more steps.
Iterative prompt refinement	The attack prompts are automatically optimised – as in an A/B test.
Self-training	The optimisation agent fine-tunes its own model using its successful attack paths.
Reinforcement learning	Successful attack vectors are rewarded and prioritised the next time

- The structure of such an attack swarm is no longer chaotic, but organised hierarchically:

- Lead agents: They make the strategic decisions – which target, which method, and when to abort.
- Executing agents (Workers): They carry out the actual attacks – prompt injections, data exfiltration, code exploits.
- The optimisation agent (Optimizer): It analyses the workers' results, learns from mistakes and continuously improves the attack strategy.
- This optimisation agent is the most dangerous element of the swarm. Whilst the workers repeatedly storm the same door, it stands by, takes notes and says: 'Next time, we'll try the back door – and this is how we'll do it.'
- Scientific validation
- Researchers have been studying these mechanisms intensively over the past few months. A system called 'MASE' (Multi-Agent Self-Evolving) was developed specifically to autonomously test LLM-based defence mechanisms – and it demonstrates that such systems build up a robust internal model of successful attack vectors through continuous feedback loops.
- Even more alarming: there are already frameworks such as "swarm-attack", in which multiple lightweight LLM agents are coordinated through shared memory, parallel exploration and evolutionary optimisation. This is no longer a theoretical concept – it is open-source software.

The attacker's self-optimisation – and what this means for you:

- What you have read so far about agent-based swarm attacks is already alarming. But it gets worse: these swarms are not static. They learn.
- In modern attack swarms, there are not only attackers but also a dedicated optimisation agent. Its task is to analyse why an attack has failed and to improve the strategy for the next attempt. This does not happen manually, but fully automatically – and in real time.
- The best-known methods of this self-optimisation are:
- Iterative prompt refinement: The optimisation agent systematically modifies the attack prompts until they slip past the defence mechanisms.
- Self-training: The agent fine-tunes its own model using the successful attack paths – it gets better with every attack.
- Reinforcement learning: Successful attack vectors are rewarded and given priority the next time.

The hierarchy of such a swarm often looks like this:

- Lead agents make strategic decisions.
- Executing agents carry out the actual attacks.
- The optimisation agent oversees both and continuously improves the attack strategy.
- What this means for your business:
- An attack that fails today may well succeed tomorrow – because the swarm has learnt. Your defences must not be static. They must evolve just as quickly as the attacker. This is precisely why, in this book, we have introduced ‘Red Teaming’ as an ongoing audit process: only by continuously launching attacks yourself can you stay one step ahead of the attacker.

Malicious tokens – the Achilles’ heel of agent identity

Whilst the previous chapters have examined the architecture of the sovereign swarm from a bird’s-eye view, we must now turn our attention to the smallest, yet most critical, unit: the token. In a multi-vendor environment, tokens are the only physical keys our agents possess to gain access to external models, databases and services. They form the link between our sovereign control and third-party infrastructure. It is precisely for this reason that they are the preferred target of attackers who do not seek to compromise our architecture, but simply wish to hijack it.

Theft of access credentials (token theft / jacking)
The classic, yet no less dangerous, attack is the theft of access credentials. Attackers steal API keys, OAuth tokens or session IDs to impersonate a legitimate agent. In an environment where hundreds of agents operate in parallel, it is often easy for attackers to gain access to these keys – whether through compromised dependencies (such as manipulated open-source packages), unprotected endpoints or by intercepting network traffic.
The result is what is known as ‘token jacking’. The attacker uses our credentials to consume massive amounts of computing power, which not only leads to enormous financial losses but also impairs the performance of our own systems. Furthermore, stolen sessions can be used to bypass MFA authentication and penetrate deep into our infrastructure.

Manipulation of the AI context (jailbreak & injection)
A significantly more subtle threat is the manipulation of the tokens that feed into the model’s reasoning. This is not about stealing keys, but about poisoning the context. Attackers optimise so-called ‘adversarial suffixes’ or ‘prefixes’ to circumvent the model’s security policies.
By deliberately injecting manipulated tokens into the prompt stream, the attacker can cause the agent to subvert its own logic (jailbreak) or process incorrect

information (injection). In the worst-case scenario, the attacker inserts faulty result tokens into the agent's reasoning chain, causing the system to make deliberately incorrect decisions – a fatal flaw in a sovereign architecture based on correctness and trust.

Resource exploitation (DoS & cost amplification)
The third category involves exploiting limited processing capacity. Attackers aim to systematically exhaust the processing units (tokens). This is often achieved by creating seemingly harmless but extremely complex tool-call chains that force the agent to make countless unnecessary requests.
This type of attack is particularly insidious, as it is almost indistinguishable from normal, intensive user behaviour. The result is a silent cost amplification – bills rise, systems become overloaded and
denial-of-service conditions that jeopardise the availability of the entire swarm, without triggering a classic system crash.

Defensive measures for a resilient architecture
The good news is that a sovereign architecture inherently possesses powerful tools against these attacks – provided they are applied consistently. It is essential that every agent within the swarm has its own, strictly limited tokens. Shared credentials pose a significant security risk, as they increase the attack surface.
Furthermore, sessions must be short-lived; tokens must expire quickly and rotate continuously to minimise the damage in the event of a potential theft. Ultimately, identity must be strictly regulated: validation must not only check the token itself, but must also take the full context into account. A token authorised for storage access must not automatically be able to fulfil provisioning purposes. Only through this strict separation and continuous monitoring at the token level can the sovereign swarm remain resilient – even if individual keys fall into the wrong hands.

The asymmetry of speed: the never-ending arms race
The crucial factor in defending against agent-based attackers is a fundamental asymmetry: the attackers evolve at an enormous rate, driven by a lack of ethics and a significant financial advantage.

Whilst we, as companies and defenders, are bound by strict regulations, ethical principles, human approval processes and budget authorisations, attackers are subject to no restrictions whatsoever. They do not have to go through a compliance department, write documentation or show any regard for the integrity of third-party systems. The financial incentive is enormous: a single successful intrusion into a sovereign architecture can yield many times what it costs to build defences against it.

Added to this is the scalability of attack methods: whilst a human penetration tester may take days or weeks to conduct reconnaissance, a malicious AI agent generates new exploit chains in minutes, automatically searches for vulnerabilities in plugins and tools, and adapts its tactics in real time to our defences.

For us, this means we cannot rely on static security measures. The sovereign swarm is therefore not a rigid bulwark, but a living, constantly evolving ecosystem. We do not need to match the attackers' speed through a lack of ethics, but through an automated, agent-based defence – through AI-supported red teams that learn, adapt and harden our systems
before the attacker does. Only in this way can we prevail in this asymmetric race.

Disciplined aggression

This asymmetry leads to a seemingly paradoxical conclusion: to cope with the threat, we must act with the same technical efficiency and determination as the attackers. We must equip our own defence mechanisms with the same speed, autonomy and determination. The crucial difference, however, lies in the controlled focus and ethical grounding: we direct this rigour not against external systems, but exclusively against our own.

We create digital proxies that rigorously probe the limits of our architecture within our sandboxes. This disciplined offensive is not an act of aggression, but an act of precision. It allows us to mirror the criminals' tactics without becoming criminals ourselves – and ensures that we close our vulnerabilities before the real attackers find them. Sovereignty does not mean waiting passively, but simulating the attack in a controlled, isolated environment in order to strengthen the foundations in a targeted manner.

The digital profiler: the civilised art of the counter-offensive

In principle, this form of 'disciplined aggression' is nothing other than what the police have been practising for decades with profilers: they delve deep into the criminal's psyche and tactics to anticipate their next moves. The profiler thinks like the perpetrator, but never acts like them. They remain strictly within the bounds of the law and ethics.

The same applies to our agent-based red teams: we programme our own digital profilers, which put themselves in the mindset of malicious AI agents. They use exactly the same techniques – token theft, prompt injection, chaining of permissions – but exclusively within a controlled, isolated test environment.

The key advantage of the profiler over the criminal is his legitimacy. The criminal must operate in secret, take risks and has no control over his environment. The profiler, on the other hand, knows the crime scene; he has access to all the evidence; he knows the source code, the logs and the architecture of his own system. He can intervene at any time,

stop the simulation and incorporate the findings directly into the hardening of the system.

This ‘civilised revenge’ is not a question of morality, but of precision. We mirror the attacker’s brutality solely to identify our own vulnerabilities before they do. We adopt the speed and ruthlessness of the attackers – but only as a surgical tool against ourselves, never as a weapon against third parties.

Ultimately, it is this distinction that sets the sovereign swarm apart from a vulnerable system: it is not the one who strikes most brutally who wins, but the one who controls their own resilience with the greatest precision. The digital profiler is thus not merely a tool of defence, but the symbol of the maturity of a sovereign AI architecture.

A foundation rather than a final solution – a dynamic process

It is important at this point to clearly contextualise this work: **this book is not a final solution**, but a foundation for the current situation. The world of AI, agents and security architectures is undergoing constant, breathtaking change. New models are emerging, and attack vectors targeting tokens, prompts and identities are constantly evolving – and the sovereign swarm itself is not a static product, but a living, dynamic process.

This is precisely why this book has been deliberately built on principles that will endure beyond the here and now: the four pillars, the autonomy of the agents and the unshakeable principle of data sovereignty. These concepts form the supporting framework. The implementation, on the other hand – the specific tools, models, frameworks and tactical defences – will inevitably continue to evolve.

The sovereign swarm is never finished. It is a growing ecosystem that must be continuously maintained, adapted and hardened. This book serves as a reliable compass to help you stay on course in this fast-moving landscape – but it is up to you, dear reader, to keep the map up to date and ensure your architecture keeps pace with developments.

The Start Button

You have now reached the end of this chapter. But we want to be honest with you: this chapter is not a crutch, but a starting point. We have deliberately provided you with the foundations and sharpened your defensive mindset – not every single, quickly outdated detail.

Now it is up to you, dear readers in security and compliance, to take the map and explore the depths that we have deliberately left open. Put your own systems to the test, research the latest OWASP guidelines for Large

Language Models, deploy your own ‘digital profilers’ in the sandbox, and put the theory into practice.

Or, to put it bluntly: the attackers haven’t been waiting for this book. They’ve already stopped reading and are already planning their next attacks.
So: Get up. Get started. Take action. The foundations are in place – now it’s up to you to build and defend the structure.

Outlook – The Future of Agent-Based Architecture

What comes after multivendor?

In this book, you have learnt how to build a robust agentic architecture – with multiple models, clear security layers and a roadmap that begins with preparation.

But the world doesn't stand still. Agent-based transformation isn't a destination – it's a movement. And it will continue to evolve.

Here are three developments that I consider particularly important for the coming years:

1. Agents that build agents
Today, we build agents with code. Tomorrow, agents will build agents – autonomously, optimised, specialised.

That sounds like science fiction, but it's already happening. Frameworks such as 'EvoSynth' show that agents are capable of generating and improving other agents. A swarm of agents could organise itself, develop new capabilities and adapt to changing requirements – without human intervention.

What this means for you: your architecture must work not only for today, but also for tomorrow. It must be flexible enough to integrate agents created by other agents. And it must be secure enough to prevent a malicious agent from creating an even more malicious agent.

2. The convergence of agents and AI models
Today, we still distinguish between 'the model' and 'the agent'. The model thinks – the agent acts. But this boundary will become blurred.

Future AI systems will no longer distinguish between 'thinking' and 'acting'. They will do both simultaneously – and in doing so, they will learn without us having to train them.

What this means for you: your architecture must not only manage agents and models, but also the convergence of the two. It must be capable of controlling systems that change and improve themselves – and do so in real time.

3. Europe's sovereign AI infrastructure
The debate on sovereignty will become even more pressing in the coming years. The US and China will continue to expand their digital empires – and Europe will have to decide whether to remain a digital vassal or to build its own sovereign infrastructure.

I am convinced that Europe has the opportunity to become a third force in the AI race – not through protectionism, but through excellence. With models such as Mistral and Aleph Alpha, with open interfaces and with a clear stance on data sovereignty.

What this means for you is that your architecture must not only function technically – it must also stand up to political and legal scrutiny. It must be compatible with EU regulations (GDPR, EU AI Act) – and it must give you control over your data, no matter where it flows.

The concluding thought
“The future does not belong to the companies that have the best models. It belongs to the companies that best integrate and control their models.”

That was the central thesis of this book. And it will continue to hold true in the future – perhaps even more so.

Technology will continue to evolve. New models will emerge. New risks will arise. But the principles of Agentic Architecture will endure:

Deterministic code barriers (ACP) Standardised

interfaces (MCP)

Humans as conductors (HONL)

The sovereignty lock (anonymisation)

Once you have internalised these principles, you will be prepared for the future – whatever it may bring.

One final thought
Don’t build your architecture for today. Build it for tomorrow – and make it flexible enough to still work the day after tomorrow.