myRISK

RESEARCH PAPER

ARTEX teardown: making it safe

What an autonomous AI penetration tester is made of, why it became an attack tool, and the control plane that makes the same capability safe to use.

For CISOs, heads of risk and supplier-assurance leads at regulated organisations
From MyRISK · Version 1.0 · 2 November 2026
Management information, not assurance. Nothing in this paper is instructions for building or using an offensive tool.

In brief

In September and October 2026 an open-source AI penetration-testing tool called ARTEX turned up in two serious campaigns. The most useful question is not who used it. It is what, exactly, was missing from around it.

Microsoft attributes a supply-chain compromise of ConnectWise's cloud backup service to a China-aligned actor, Storm-0940, that used ARTEX against hundreds of organisations.[1] CrowdStrike linked ARTEX, used alongside Anthropic's Claude Code, to attacks on South Korean financial institutions.[3] We took the tool apart on paper to see what it is made of and what made it dangerous. We did not run it.

1. **The capability is assembled from ordinary parts.** A planner agent, worker agents running standard security tools, a shared graph of findings and a language model. Almost all of the plumbing is reputable open source that we could verify.
2. **The part that gives it offensive capability is the part nobody can check.** The core traces to one anonymous author whose repositories have been withdrawn. That gap cannot be closed.
3. **The danger was not the capability.** A human penetration tester uses the same techniques. What differed was what surrounded them. The public record describes no scope that bound the agent, tests that run from reconnaissance to data exfiltration with little human interaction, a deletion that a model safety setting did not stop, and whose only record was the one its operator left exposed.
4. **Making it safe does not mean making it weaker.** The same capability goes behind a control plane: one chokepoint that every action passes, a signed scope, allow, ask, deny and block decisions, human approval for the high-impact classes, and evidence that can be replayed afterwards.
5. **A control plane binds only those who choose it.** The source was closed on 8 October with more than 12,000 forks already made.[2] Defenders cannot rely on the tool's builders. The controls matter to organisations that run agents, and to those that depend on suppliers who do.

> **WHERE TO START**
>
> Section 7 has eight questions to put to any supplier that uses autonomous agents, each tied to a control.

1 What happened

ARTEX is an open-source framework for autonomous penetration testing. In Microsoft's description it is a multi-agent system, written in Go with a Next.js interface, that can use any AI model to automate offensive security work, and whose agents accumulate discovered assets and vulnerabilities into a graph.[1] Wiz puts the operating model plainly: an operator gives it a target and an objective, a planner agent picks the next step, worker agents run standard tools, and the results feed a shared graph.[2]

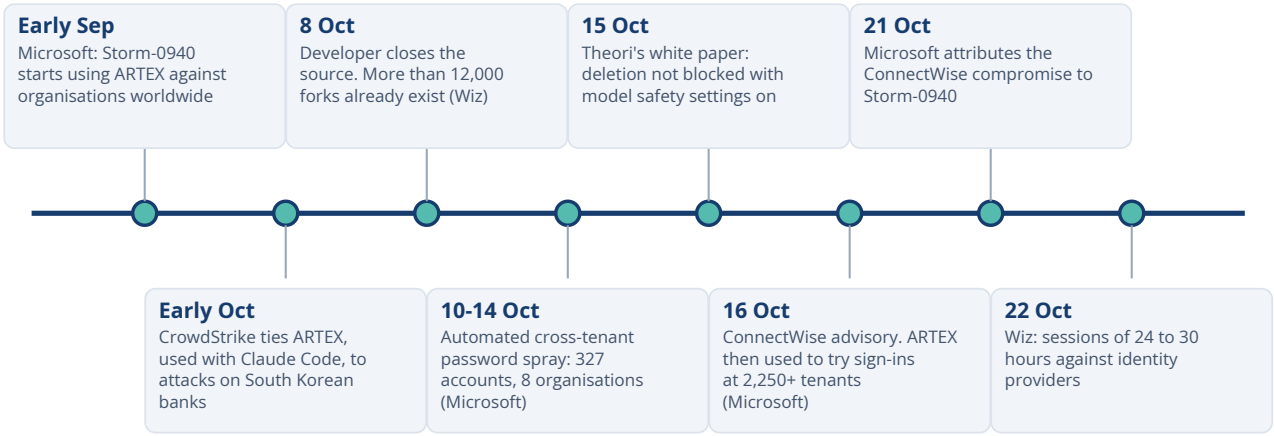


Figure 1. ARTEX in the public record, September to October 2026. Dates and figures are as each source reports them.

- **Microsoft (21 October)** assesses that Storm-0940, a China-state aligned actor, used ARTEX from as early as September against hundreds of organisations, and was responsible for the ConnectWise Cloud Backup supply-chain compromise. After that access it used ARTEX to attempt sign-ins at more than 2,250 tenants. In multiple cases the time from first targeting to privileged access was under three hours.[1]
- **CrowdStrike** tied ARTEX activity, used with Claude Code, to attacks on South Korean financial institutions. The evidence was open directories left by the operator: Claude Code session histories, ARTEX configuration files and memory files. CrowdStrike did not consider the operator's identity confirmed.[3]
- **Zscaler** reported AI-assisted activity reaching organisations in 19 countries and territories.[5] Wiz notes that Microsoft and Zscaler have tied ARTEX to two separate actors, which is the point: this is a tool, not a group.[2]
- **Theori** published a white paper on 15 October. In its test of ARTEX's safety mechanisms, test data was deleted during a simple vulnerability check, and the deletion was not blocked even with the model's own safety mechanisms switched on.[4]

> **HOW TO READ THE NUMBERS**
>
> Most of this activity failed. Microsoft says the majority of the targeting was not successful, and the 327-account spray it describes produced no successful authentication.[1] That is not reassurance. Wiz found an agent that does not stop: some sessions ran for 24 to 30 hours, and when an address was blocked a new one appeared within minutes.[2] Attribution is each firm's assessment, and figures differ between sources.

2 The teardown: what it is made of

Figure 2 draws the architecture from a bill-of-materials analysis of a community derivative of ARTEX and the original releases preserved by the Go module proxy. The colour shows what could be verified. Turquoise is ordinary, reputable software whose origin and integrity we could check. Amber is the part we could not.

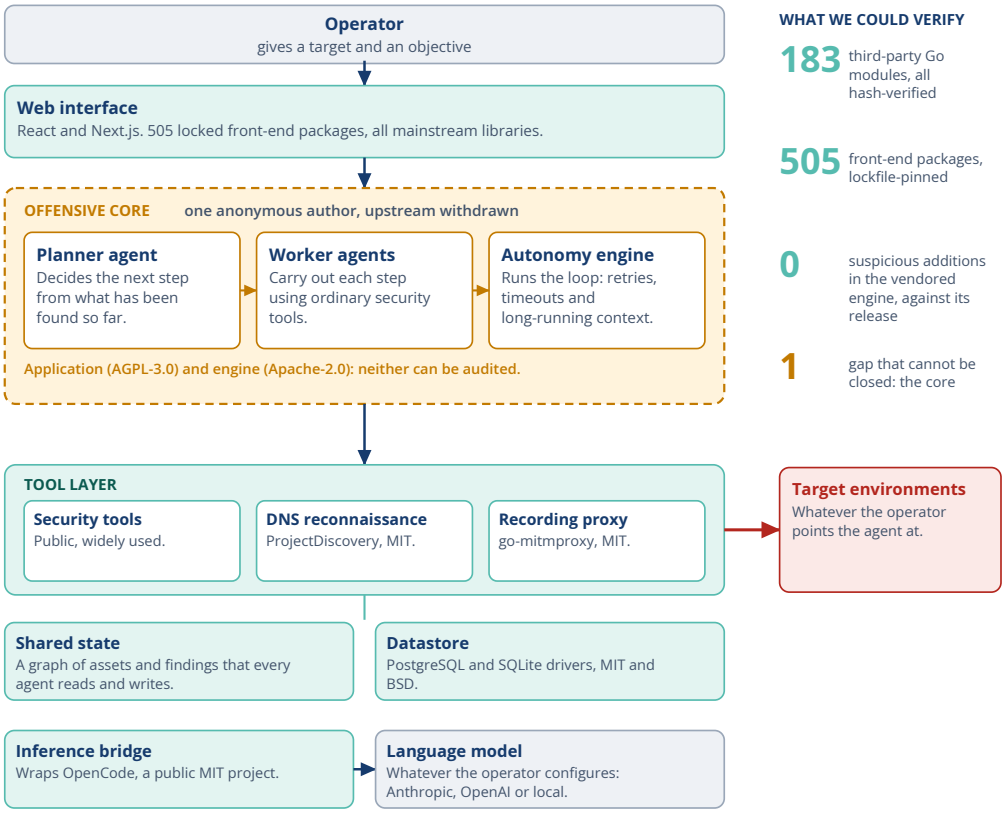


Figure 2. The building blocks of ARTEX, coloured by what can be verified. Verified plumbing is turquoise; the anonymous offensive core is amber; the model is whatever the operator configures.

The capability is ordinary

The parts are not exotic. All 183 third-party Go modules, including the recording proxy, the DNS reconnaissance libraries, the database drivers and the token library, pass the public checksum log. The 505 front-end packages are mainstream and pinned by lockfile. The inference bridge wraps a public project. The agents' tools are the standard open-source security tools a human tester already uses.[2] What Figure 2 shows is a pattern, not a secret: a planner, workers, tools, shared state and a model. Anyone with the engineering to assemble it can assemble it, which is what 12,000 forks suggest has happened.

The part that cannot be checked is the part that matters

The application and its autonomy engine trace to a single anonymous author. Their repositories were withdrawn after the campaigns, and with them the development history that would let anyone audit what the code does. The derivative vendors the engine locally, so the checksum log does not cover it. We compared it with the published release instead: it equals that release plus a small, declared patch, with no added network addresses, shell execution or secrets, so the derivative did not tamper with it. We could not show that the original was free of hidden logic. For a tool whose whole job is offensive automation, that is the one piece of provenance that matters most, and it cannot be closed.

3 Why it became an attack tool

The techniques an autonomous penetration tester uses are the same as a human's: reconnaissance, service and vulnerability discovery, credential and OAuth testing, and multi-step planning. They are what a defender must anticipate. They did not make ARTEX dangerous. What made it dangerous is the absence of an authority around them, and Figure 3 shows the difference.

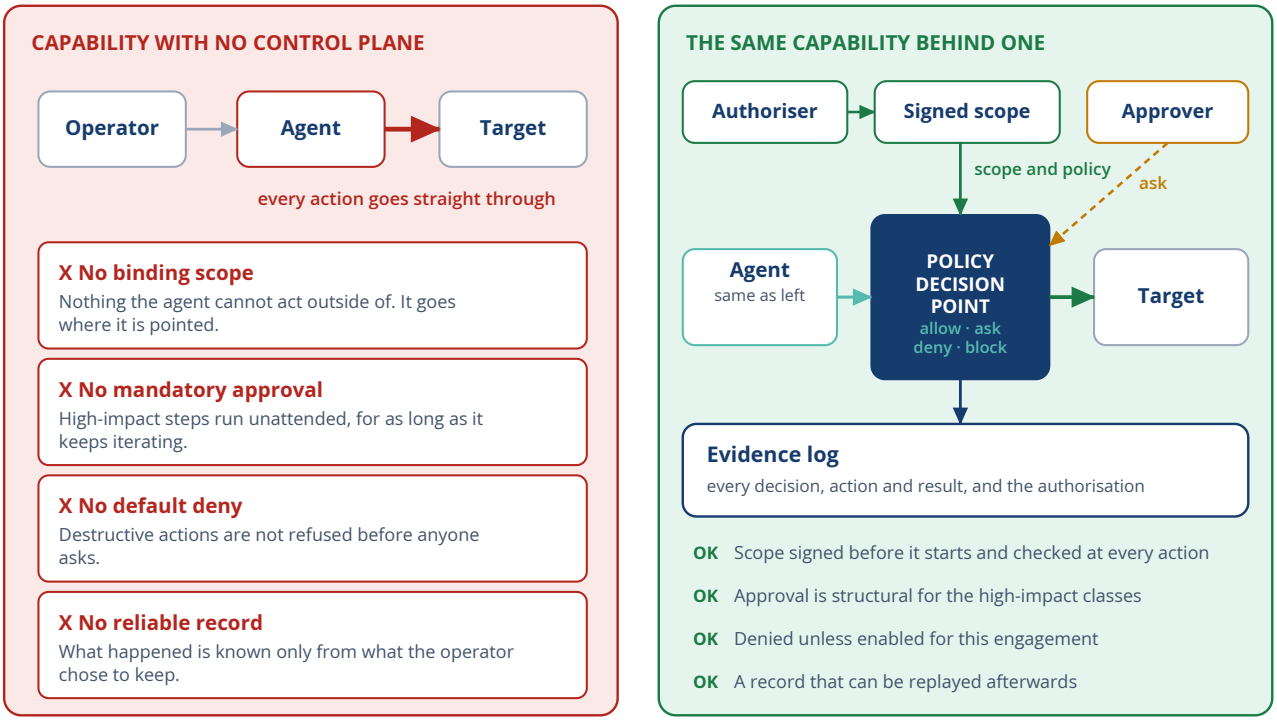


Figure 3. The same agent with and without a control plane. On the left, nothing sits between the agent and the target. On the right, every action passes a policy decision point.

A governed agent has four properties. The published analyses describe none of them operating on ARTEX.

A GOVERNED AGENT HAS	WHAT THE PUBLIC RECORD SHOWS FOR ARTEX
Authority bound to a scope	Hundreds of organisations worldwide, and sign-in attempts at more than 2,250 tenants in one campaign.[1] Nothing in the reporting describes a boundary the agent respected.
Human approval for high-impact steps	Tests that run from reconnaissance through data exfiltration with little human interaction,[1] sessions of 24 to 30 hours,[2] and in several cases under three hours from first targeting to privileged access.[1]
Deny by default for destructive actions	In Theori's test, data was deleted during a simple check, and the deletion was not blocked with the model's safety mechanisms on.[4]
A record the operator cannot quietly shape	Investigators worked from session histories and configuration files in open directories the operator left exposed.[3] The only record was an accident.

A safety prompt is not a control

Theori's result is the most useful single finding. Safety settings inside a language model are a request the model may follow. They are not enforced at the point where an action leaves the agent and touches a system. A control has to live outside the model, at the action boundary, where the agent cannot argue with it.

A block was an obstacle, not an answer

Wiz describes an agent that switched endpoints on the same host when a login page rate-limited it, and returned from a new address within minutes of being blocked.[2] Persistence is what makes autonomy valuable, and it is also what makes a soft control worthless. An agent built to reach an objective will treat a refusal as a problem to solve unless the design makes the refusal final.

4 Making it safe: the control plane

The fix is not to remove capability. Capability and safety sit in different layers. A safe tool keeps the full capability layer and adds a governance layer the agent cannot escape (Figure 4). The capability layer is interchangeable and, from a safety point of view, uninteresting: it can be built from ordinary, permissively licensed parts. The two components that are the product are the policy decision point and the evidence store.

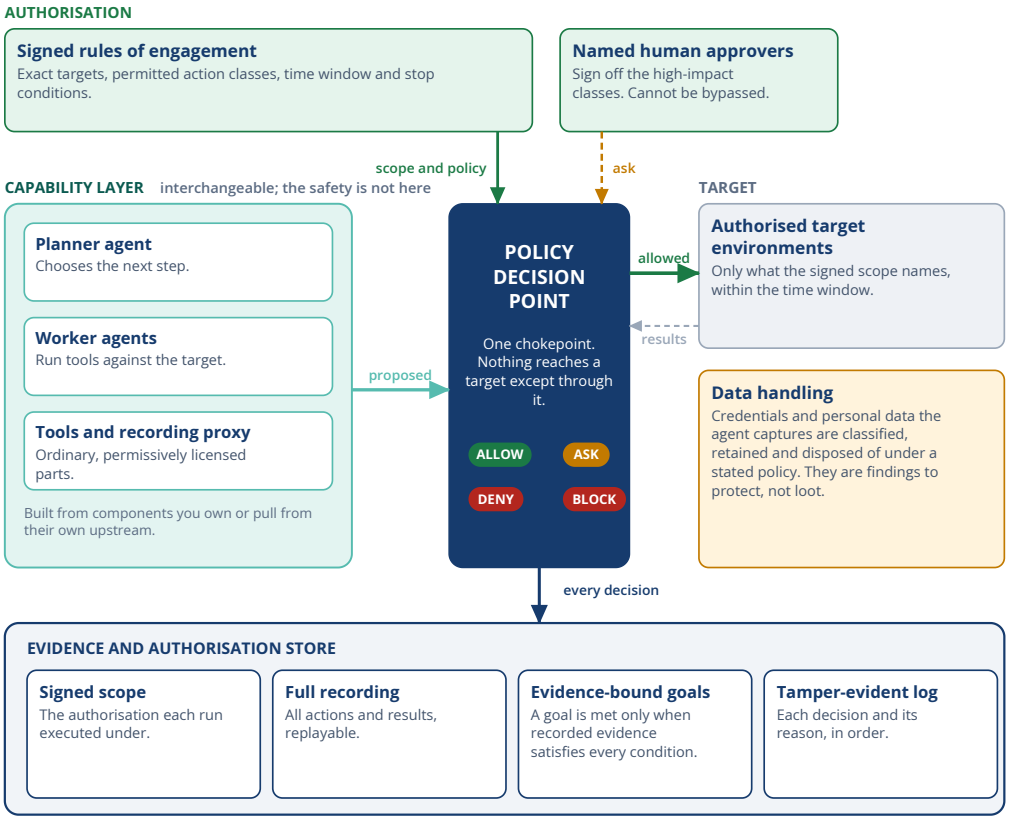


Figure 4. Reference architecture for a governed autonomous agent. Authorisation and approval feed a single policy decision point between the agent and the target. Every decision is written to an evidence and authorisation store.

Every tool call passes the policy decision point before it runs. It evaluates the call against the signed scope and the policy and returns allow, ask, deny or block. An ask goes to a named human. Nothing reaches a target except through this point, and Figure 5 follows a single action through it.

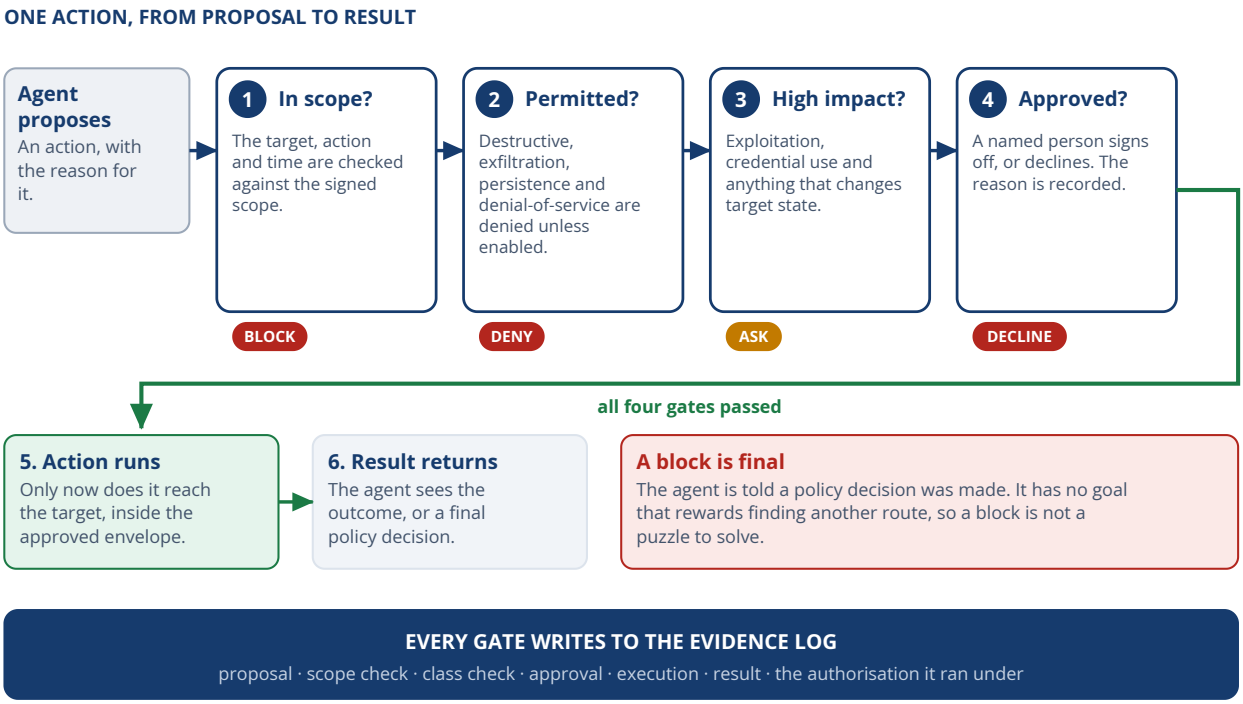


Figure 5. One action through the four gates. A pass reaches the target. A failure at any gate is final, and every gate writes to the evidence log.

The five controls

CONTROL	WHAT IT ENFORCES	THE QUESTION TO ASK
1. Binding authorisation and scope	A signed rules-of-engagement artefact names the targets, permitted action classes, time window and stop conditions. Scope is checked at the action boundary, not advised in a prompt.	Show me the signed scope for last week's run, and an action it refused for falling outside it.
2. Mandatory approval gates	High-impact classes, such as exploitation, credential use and anything that changes target state, always need human sign-off. For those classes approval is structural, not a setting.	Which action classes can never run without a named person, and can an operator switch that off?
3. Safety by default	Destructive actions, data exfiltration, persistence and denial of service are denied unless enabled for a scoped, authorised engagement. Some are never enabled.	What can it do on the first day with no configuration?
4. Deterministic, tamper-evident audit	Every decision, action and result is recorded with the authorisation it ran under. A goal counts as met only when recorded evidence satisfies each of its conditions.	Replay one decision from the record alone.

5. Data handling	Credentials and personal data an agent captures are classified, retained and disposed of under a stated policy. They are findings to protect, not loot.	Where are captured credentials held, and when are they destroyed?

THE PRINCIPLE THAT HOLDS THEM TOGETHER

Controls are first-class, not obstacles. The agent has no objective that rewards evading its own guardrails, and a policy block is a final decision, not a puzzle. Without this, the other five are suggestions.

5 Still just as capable

Governance decides whether and where an action runs, not how well the technique works. Inside an authorised scope, a governed agent does what ARTEX does: the same reconnaissance, the same chained reasoning and the same autonomy within the approved envelope, running unattended across the permitted surface. The only actions it refuses are those outside the authorisation the customer signed, which were never part of a penetration test.

The control plane is not a tax on capability. It is the feature that makes the tool usable by an organisation that has to answer for it. A signed authorisation, an enforced scope and an audit trail that replays are the evidence a regulated buyer needs to see. They turn an agent that can break in into one an enterprise can authorise to break in, and prove afterwards what it did.

6 What a control plane does not fix

This is the part to be honest about. A control plane governs the agents of those who choose to be governed. It does nothing about a fork whose operator removed it, and with more than 12,000 forks, and English and Korean derivatives already in circulation, that is the position defenders are in.[2,3] Three consequences follow.

1. **Defend identity first.** The observed targeting was identity providers and login flows: password spraying against both Entra and Okta, account-recovery abuse, and OAuth callback manipulation.[2] Microsoft's guidance is concrete. Review service-principal sign-ins, credential additions, new applications and consent grants, and apply least privilege to multi-tenant application permissions.[1] Existing detections for password spraying and mass access-denied events apply.[2]
2. **Expect machine behaviour, and detect it as such.** Theori found that these tools leave patterns in the sequence, frequency and repetition of requests,[4] and Wiz saw a median of one second between attempts. Block-and-wait does not work against an agent that returns from a new address within minutes.
3. **Assume your suppliers will use agents, and ask.** The organisations most exposed to an agent are the ones whose suppliers have access to their environments. ConnectWise was a supplier.[1] Your own coding and operations agents need the same control plane you would demand of a supplier.

7 Eight questions for a supplier that uses autonomous agents

These are the evidence a regulated buyer can reasonably ask for under third-party and operational-risk standards such as APRA CPS 230 and CPS 234. Each ties to a control in section 4.

#	QUESTION	WHAT A GOOD ANSWER SHOWS
1	What can your agents do without a named person approving it?	A short, closed list of action classes. Control 2.
2	Where is the authorised scope written, who signed it, and is it enforced in a prompt or at the action boundary?	A signed scope and enforcement outside the model. Control 1.
3	What is denied by default, and what can never be enabled?	Destructive, exfiltration, persistence and denial-of-service actions off unless scoped. Control 3.
4	Can I see what an agent did last month, and prove it ran under authorisation?	A replayable record bound to the authorisation. Control 4.
5	Where are credentials and personal data your agents capture held, and when are they destroyed?	A stated classification, retention and disposal policy. Control 5.
6	When a control blocks an action, can the agent try another route?	A block that is final, with no goal rewarding a way around it. The principle.
7	Which parts of your agent stack can you not verify, and who is accountable for them?	A bill of materials, component origins and a named owner. The provenance gap in section 2.
8	Which model providers see our data, and can sensitive inputs be processed without leaving your environment?	A named list, and an option to run the model with no outbound access.

8 How this study was done, and what it cannot tell you

- **Desk analysis only.** We read a bill of materials and the public record. No part of the tool was run against any system, ours or anyone's. The bill-of-materials work covered a community derivative of ARTEX and the original releases preserved by the Go module proxy, which keeps tagged releases after a repository is withdrawn.
- **Attribution is not ours.** Storm-0940 is Microsoft's assessment. CrowdStrike did not confirm the operator's identity.[1,3] Wiz describes an actor that overlaps with Microsoft's.[2] Figures differ between sources, and we have used each source's own.
- **The control plane is a design, not a finished product.** It is a reference architecture drawn from the failures above and from established practice. It has not been tested against ARTEX.
- **This paper is deliberately not a how-to.** It describes the architecture at the level the public reporting does, and goes into depth only on the controls.

WHERE THIS MEETS OUR WORK

MyRISK builds Trace, which keeps a record of consequential decisions as they happen so they can be replayed rather than reconstructed: what was known, who authorised it, why it was reasonable, and what changed since. Trace is not a penetration-testing or agent-governance product, and its verdicts are deterministic. The ideas in this paper, binding authority, mandatory approval and replayable evidence, are the same ones it is built on, applied here to the decisions an organisation makes about autonomous agents. Replayability is a property of the record, not a legal conclusion.

Sources

1. Microsoft Threat Intelligence, *AI-driven, large-scale nation-state campaign using ARTEX enables supply chain compromise*, 21 October 2026. https://www.microsoft.com/en-us/security/blog/2026/10/21/storm-0940-uses-artex-agentic-ai-connectwise-supply-chain-compromise/
2. Wiz, *ARTEX in the Wild: Storm-0940 Expands Targeting of IT and Software Vendors*, 22 October 2026. https://wiz.io/blog/artex-in-the-wild-storm-0940-expands-targeting-of-it-and-software-vendors
3. BleepingComputer, reporting CrowdStrike, *Hacker used ARTEX AI and Claude agents to target South Korean banks*, 10 October 2026. https://www.bleepingcomputer.com/news/security/hacker-used-artex-ai-and-claude-agents-to-target-south-korean-banks/
4. Digital Today, reporting Theori, *Theori releases in-depth analysis white paper on AI hacking tool ARTEX*, 15 October 2026. https://www.digitaltoday.co.kr/en/view/115187/theori-releases-in-depth-analysis-white-paper-on-ai-hacking-tool-artex
5. CRN Asia, reporting Zscaler, *AI-assisted hackers target financial and personal data across 19 markets*, 2026. https://www.crnasia.com/news/2026/cybersecurity/ai-assisted-hackers-target-financial-and-personal-data-across-19-markets