

Agent Sandboxing Principles Matrix

Six First Principles of Agent Sandboxing. How each sandbox type meets them, the controls that put them into practice, the research behind them, and examples of recent incidents that demonstrate violations of these principles.

1 Isolation Every agent gets its own room, with walls to the ceiling and its own share of the heat, and the room is emptied before the next guest arrives.	2 Deny by default The room is built with no doors at all, so an agent with zero capabilities can't touch anything, even by accident.	3 Least authority You cut only the doors the job needs, and each one opens into a single room, never into a hallway that reaches every room.	4 Policy Enforcement Points Every door gets a guard who checks each trip through it, and checks which room you actually enter, not what the sign on the door says.	5 Controlled information flow The guard also checks what you carry out, because an allowed door can still be used to carry out the wrong papers.	6 Composition Rooms connect only through guarded doors, so you can add a new room without re-inspecting the whole building.
---	---	---	---	---	--

Part A · Sandbox types against the principles

Rating is the default posture; OS sandbox columns are rated as configured. Pro, Con, and the controls to add.

● **Yes** meets it by default ● **Partial** partly, or with the listed controls ● **No** not by default Columns ranked by their counts across all 15 rows (Yes = 2 points, Partial = 1), strongest first.

OS sandbox = bubblewrap / Landlock / seccomp / cgroups (Linux) · Seatbelt (macOS) · AppContainer / Job Objects (Windows). gVisor = the `runsc` user-space kernel.

PRINCIPLE	Wasm	OS sandbox Linux	OS sandbox macOS	OS sandbox Windows	microVM	gVisor	Container	VM
	9 Yes 6 Partial 0 No	7 Yes 8 Partial 0 No	4 Yes 10 Partial 1 No	3 Yes 12 Partial 0 No	3 Yes 10 Partial 2 No	1 Yes 13 Partial 1 No	1 Yes 13 Partial 1 No	2 Yes 10 Partial 3 No
1 Isolation Share nothing: not memory, not budgets, not leftovers. Each sandbox has a clear edge in space (memory and kernel), in resources (CPU, memory, slots, descriptors, disk) and in time (nothing left behind for the next tenant). Every agent gets its own room, with walls to the ceiling and its own share of the heat, and the room is emptied before the next guest arrives.	● Partial PRO Own memory per instance; no OS needed; microsecond start CON Software-enforced; instances share a process and the compiler ADD - One instance per request - Pooling allocator with guard pages - Spectre mitigations - Run the host inside a microVM for a second wall	● Partial PRO Namespaces + cgroups give each sandbox its own view and its own budget, cheaply CON One shared kernel: a kernel bug breaks every sandbox ADD - bubblewrap (user, mount, pid, net namespaces) - cgroups v2 for budgets - gVisor for a user-space kernel	● Partial PRO Seatbelt confines files, network and IPC per process CON Shared kernel; no per-sandbox budgets beyond rlimits ADD - Seatbelt profile (<code>sandbox-exec</code>) - Apple <code>container</code> (VM per container) for a hardware wall	● Partial PRO AppContainer gives its own object namespace and token; Windows Sandbox adds a VM CON AppContainer shares the kernel; budgets need Job Objects ADD - AppContainer / LPAC - Job Objects for CPU and memory caps - Windows Sandbox (Hyper-V)	● Partial PRO Strong hardware-enforced wall with a tiny device model; ~125 ms boot CON Not functional on its own: needs a guest OS (kernel + rootfs), and KVM on a Linux host ADD - Firecracker jailer - Keep optional devices (PCI) off - One microVM per task	● Partial PRO A user-space kernel (the Sentry) sits between the app and the host kernel; much stronger than a plain container CON Software boundary over a shared host kernel, narrowed by seccomp; Linux only ADD <code>runsc</code> as the OCI runtime - systrap or KVM platform - GKE Sandbox / Kubernetes RuntimeClass	● Partial PRO Own filesystem, process and network namespaces; fast CON Shares the host kernel (runc, NVIDIA toolkit escapes) ADD - Rootless / user namespaces - gVisor (<code>runsc</code>) or Kata Containers - No host paths, no <code>docker.sock</code> - Pod Security Standard: restricted	● Partial PRO Strongest isolation available: a hardware-enforced wall (VT-x / AMD-V, nested page tables) with its own kernel CON Not functional on its own: needs a full guest OS to run anything, and that guest is wide open inside; slow to boot ADD - Disable shared folders and clipboard - Patch the hypervisor - Confidential VM if the host is untrusted
1a Memory & kernel The wall itself: separate address spaces, kernels or linear memories, so one tenant can't read or write another's state.	● Yes PRO Bounds-checked linear memory per instance; no kernel exposed. Cranelift formal verification checks the instruction-selection rules that turn Wasm memory accesses into machine code CON Software-enforced; compiler paths outside the verified rules can still be escapes; shared process ADD - Pooling allocator with guard pages - Spectre mitigations - Second wall: OS sandbox or microVM	● Partial PRO PID, mount and user namespaces hide other processes and files CON The shared kernel is the wall ADD - bubblewrap <code>--unshare-all</code> - seccomp to shrink the kernel surface - gVisor	● Partial PRO Seatbelt blocks process inspection and Mach lookups CON Shared XNU kernel ADD - Seatbelt deny <code>process-info*</code>, <code>mach-lookup</code> - Hardened Runtime	● Partial PRO AppContainer blocks handles to other processes and objects CON Shared NT kernel ADD - AppContainer / LPAC - Windows Sandbox for a separate kernel	● Partial PRO Strong hardware wall with a tiny device model CON Not functional on its own: needs a guest OS and KVM ADD - Firecracker jailer - Keep optional devices (PCI) off	● Partial PRO App syscalls go to the Sentry, not the host kernel; the Sentry itself runs under a tight seccomp filter CON A Sentry bug is an escape into a seccomp-confined process, not past a hardware wall ADD - KVM platform for hardware-backed address spaces - Keep <code>runsc</code> patched	● Partial PRO Own namespaces CON Shares the host kernel (runc, NVIDIA toolkit escapes) ADD - Rootless / user namespaces - gVisor (<code>runsc</code>) or Kata - No host paths, no <code>docker.sock</code>	● Partial PRO Strongest wall available: hardware-enforced, its own kernel CON Not functional on its own: needs a full guest OS ADD - Disable shared folders and clipboard - Patch the hypervisor - Confidential VM if the host is untrusted
1b Resource budgets Each tenant gets a fixed share of CPU, memory, instance slots, file descriptors, connections and disk, so no tenant can starve the rest.	● Partial PRO Runtime can meter fuel, interrupt by epoch and cap memory per store CON Budgets are runtime features you must switch on; shared pools (instances, descriptors) need per-tenant partitioning ADD - Fuel + epoch interruption <code>ResourceLimiter</code> memory caps - Per-tenant instance and connection quotas	● Yes PRO cgroups v2 cap CPU, memory, pids (fork bombs) and I/O per sandbox CON bubblewrap sets no budgets itself; pair it with a cgroup. Inodes and page cache stay shared ADD - cgroups v2 (<code>systemd-run --scope</code>) <code>pids.max</code> - Size-limited tmpfs	● Partial PRO rlimits and QoS classes CON No per-sandbox CPU or memory caps comparable to cgroups ADD <code>setrlimit</code> - launchd resource limits - A VM for hard caps	● Yes PRO Job Objects cap CPU rate, memory and process count CON AppContainer alone sets no budgets; needs a Job Object ADD - Job Objects - Windows Sandbox memory settings	● Yes PRO Fixed vCPUs and RAM; built-in disk and network rate limiters CON Overcommit at density brings noisy neighbors back ADD - Firecracker I/O rate limiters - cgroups via the jailer	● Yes PRO Runs under the container's cgroups; the Sentry accounts memory per sandbox CON Kernel-wide host resources (page cache, conntrack) stay shared ADD - Pod CPU / memory limits - RuntimeClass overhead settings	● Yes PRO cgroup CPU, memory, pids and I/O limits per container; Kubernetes quotas CON Kernel-wide resources (inodes, conntrack, page cache) stay shared ADD - Pod CPU / memory limits <code>--pids-limit</code> - ResourceQuota / LimitRange	● Yes PRO Fixed vCPUs and RAM per VM CON Host overcommit brings noisy neighbors back ADD - No memory overcommit for hostile tenants - CPU pinning
1c Residue & reuse Nothing outlives the tenant. Scratch files, memory, pooled connections, cached sessions and reused identifiers are wiped or retired before anything else gets them. This is the Orange Book's object reuse requirement.	● Partial PRO Fresh instance per request; pooled memory is reset before reuse CON Preopened directories and host-side pools outlive the instance ADD - Instance per request - Host-owned scratch with reclaim - Never reuse tenant IDs	● Yes PRO <code>--tmpfs</code> gives each run a fresh scratch space that vanishes on exit CON Bind-mounted workspaces persist by design ADD - bubblewrap <code>--tmpfs /tmp</code> - systemd <code>PrivateTmp</code>	● Partial PRO Per-app container directories CON Container directories persist; no ephemeral scratch per run ADD Per-run temp directory, deleted on exit	● Partial PRO Windows Sandbox discards all state on close CON AppContainer storage persists between runs ADD Windows Sandbox for disposable runs	● Partial PRO Cheap to throw away per task CON Snapshot restore brings back old memory, secrets and RNG state ADD - One microVM per task - Verify snapshots before restore - Reseed randomness after restore	● Partial PRO Sandbox state lives in the Sentry and is discarded with it CON Checkpoint and restore bring back old memory and secrets; volumes persist ADD - Ephemeral sandbox per task - Verify checkpoints before restore	● Partial PRO Writable layer is discarded with the container CON Volumes and scratch need explicit cleanup ADD <code>--rm</code> <code>--read-only</code> + tmpfs - Size-limited emptyDir	● Partial PRO Discard the disk to wipe everything CON Long-lived VMs accumulate state; snapshots capture secrets ADD Ephemeral VM per task

PRINCIPLE	Wasm	OS sandbox Linux	OS sandbox macOS	OS sandbox Windows	microVM	gVisor	Container	VM
	9 Yes6 Partial0 No	7 Yes8 Partial0 No	4 Yes10 Partial1 No	3 Yes12 Partial0 No	3 Yes10 Partial2 No	1 Yes13 Partial1 No	1 Yes13 Partial1 No	2 Yes10 Partial3 No
2 Deny by default Start with nothing. Access exists only where it's explicitly wired in. No auto-approve, no auto-start, and no denylists standing in for safety. The room is built with no doors at all, so an agent with zero capabilities can't touch anything, even by accident.	Yes PRO No imports means no access. Needs are declared in the component's imports, not guessed from its behavior CON Every capability needs host wiring ADD Grant each WIT import deliberately	Yes PRO bubblewrap starts from an empty filesystem view; Landlock and seccomp support deny-all CON Rarely used: by Rice's theorem you can't know in advance what an arbitrary program will need, so few tools ship a deny-all profile ADD - bubblewrap <code>--unshare=all</code> - Landlock deny-all + grants - seccomp allowlist	Yes PRO Seatbelt profiles can start (<code>deny default</code>) CON Rarely used: the profile has to predict what a program will need, which Rice's theorem says can't be done in general; the language is undocumented too ADD - Seatbelt (<code>deny default</code>) - App Sandbox entitlements	Yes PRO AppContainer starts with no capabilities CON Rarely used: predicting every capability an arbitrary program needs is undecidable (Rice's theorem), and few CLI tools support it ADD - AppContainer / LPAC - App Control allow mode	Partial PRO Only attached devices exist CON The guest OS is wide open ADD No network device unless needed	Partial PRO The Sentry's own access to the host is a fixed seccomp allowlist CON Inside the sandbox the app sees a normal Linux with ambient authority; files and network follow container defaults ADD <code>--network none</code> <code>--cap-drop ALL</code>, read-only rootfs	Partial PRO Default seccomp blocks ~44 risky syscalls CON Network and capabilities stay broad ADD <code>--cap-drop ALL</code> <code>--network none</code> <code>--read-only</code>, <code>no-new-privileges</code>	Partial PRO Host is denied by default CON Inside the guest, everything is allowed ADD No NIC or shared folders until needed
3 Least authority Give each part only the authority its task needs, and make every grant an explicit, unforgeable capability: a reference that can be handed over but never guessed or fabricated. Authority lying around (your account, env vars, a shared token) is ambient, and ambient authority is how excess authority gets used. You cut only the doors the job needs, and each one opens into a single room, never into a hallway that reaches every room.	Yes PRO Per component, per resource. Each grant is an unforgeable handle the host passes in; a component can't guess or fabricate one CON Only as narrow as the interface design ADD Narrow WIT interfaces (one bucket, one host)	Partial PRO Grant one directory, one port, one socket CON Granularity is paths and ports, not resources or calls ADD - Landlock per-path rules - Separate UID - Drop capabilities	Partial PRO Per-path and per-service rules CON Network rules can't name domains ADD - Seatbelt per-path rules - Scoped Keychain items	Partial PRO Capabilities and brokered file access per app CON Capabilities are coarse (e.g., all internet) ADD - LPAC with minimal capabilities - Low integrity level	Partial PRO Cheap enough for one per task CON Still per-machine inside the guest ADD One microVM + one scoped token per task	Partial PRO Per-workload identity is possible, as with containers CON Service accounts and mounts are often too broad ADD - Per-workload identity - Least-privilege mounts	Partial PRO Per-workload identity is possible CON Service accounts are often too broad ADD - Per-workload identity - Least-privilege service accounts	Partial PRO Scoped instance role per VM CON Authority per machine is coarse ADD - Scoped instance roles - One VM per trust level
4 Policy Enforcement Points (complete mediation) Put an enforcement point in front of every resource: every access crosses it, nothing goes around it, and it is small enough to verify. It checks the real resource (the resolved file, the real destination), not its name. The term comes from NIST's zero-trust architecture, where a Policy Enforcement Point (PEP) enforces what a Policy Decision Point (PDP) decides; the idea is Anderson's reference monitor. The sub-rows are the PEPs a sandbox needs. Every door gets a guard who checks each trip through it, and checks which room you actually enter, not what the sign on the door says.	Partial PRO Every import call and memory access is checked CON The compiler is in the trusted base ADD - Keep the host interface small - Fuzzed compiler (Cranelift); stay patched	Partial PRO LSM hooks and seccomp see every syscall CON The kernel is the trusted base ADD - Landlock - seccomp-bpf (kernel surface) - AppArmor / SELinux	Partial PRO The sandbox kernel extension checks every operation CON XNU is the trusted base; Seatbelt is deprecated ADD - Seatbelt - SIP - TCC	Partial PRO The Security Reference Monitor checks every object access CON Huge trusted base ADD - AppContainer - App Control - VBS / HVCI	Partial PRO Hardware-trapped mediation through a small VMM with a handful of virtio devices CON Mediates only the guest-host edge; inside, the guest kernel decides ADD Jailer: seccomp, cgroups and chroot around the VMM	Partial PRO Every app syscall is handled by the Sentry, written in memory-safe Go, before anything reaches the host CON The Sentry reimplements much of Linux, so it is a large trusted base of its own ADD <code>runsc</code> with systrap or KVM - Keep the host seccomp filter tight	Partial PRO seccomp and LSM profiles on by default CON The guard is the shared host kernel ADD - Custom seccomp profile - gVisor puts a smaller guard in front of the kernel	Partial PRO Hardware traps every privileged instruction and device access to the hypervisor CON Mediates only the guest-host edge; inside, the guest OS decides. Large device emulation (e.g., QEMU) is trusted ADD - Minimal virtual hardware - Sandbox the VMM process
4a Filesystem Which files can be read, written or executed. Check the resolved path (after symlinks and ..), not the string, and enforce it below the agent's tools.	Yes PRO No filesystem unless a directory is preopened CON Granularity is the directory ADD - Preopen only the workspace subdirectory - Read-only where possible - WASI-Virt for a virtual filesystem	Yes PRO Landlock enforces per-path rules in the kernel with no root; bubblewrap binds only the workspace CON Resolve symlinks before granting ADD - Landlock ruleset - bubblewrap <code>--ro-bind / --bind</code> <code>noexec</code> mounts	Yes PRO Seatbelt file rules by subpath CON Profile language is undocumented ADD Seatbelt <code>deny file-read*</code> except the workspace	Partial PRO AppContainer blocks files unless ACL'd or brokered CON Granting a folder means editing ACLs ADD - AppContainer ACLs - Brokered file pickers	Yes PRO Own block device CON Files in and out need a sync step ADD Workspace on a separate, disposable drive	Partial PRO File access goes through the Sentry (via the Gofer, or seccomp-restricted directfs) CON Bind mounts still expose host files ADD - Mount only the workspace - Read-only rootfs	Partial PRO Own root filesystem CON Bind mounts leak host files ADD - Read-only rootfs - Mount only the workspace	Yes PRO Own disk CON Shared folders undo it ADD No shared folders, or read-only ones
4b Network egress allowlist Where the sandbox can send data. Allow specific destinations and also check method, path and credential: an allowed domain can still reach an attacker's account.	Yes PRO No network unless imported; host sees each request's method and URL CON The host has to implement the allowlist ADD - Grant <code>wasi:http</code> only - Enforce the destination allowlist in the host handler	Partial PRO An empty network namespace means no network; a proxy on a Unix socket gives a domain allowlist CON The kernel can't filter by domain; needs a proxy ADD <code>bwrap --unshare-net</code> + proxy - nftables - Landlock TCP ports	Partial PRO Seatbelt can deny all network except a local proxy CON No domain filtering in the profile; needs a proxy ADD - Seatbelt <code>deny network*</code> + proxy - Network Extension filter	Partial PRO AppContainer without <code>internetClient</code> has no network; WFP per-app rules CON Rules are IP/port; domain allowlists need a proxy ADD - AppContainer capabilities - WFP rules	Partial PRO No network unless you attach a tap CON Domain rules need host nftables + a proxy ADD - tap + host nftables + egress proxy - vsock only for control traffic	Partial PRO netstack is a user-space TCP/IP stack, so the host network stack isn't exposed CON Egress is open by default, as with containers ADD <code>--network none</code> - NetworkPolicy / Cilium egress rules	Partial PRO Own network namespace; Cilium adds DNS-aware and L7 rules CON Egress open by default ADD <code>--network none</code> - Kubernetes default-deny NetworkPolicy - Cilium <code>toFQDNs</code> + L7 HTTP policy	No PRO One virtual NIC to police CON NAT gives full internet; metadata service reachable ADD - Host firewall or egress proxy - Block the cloud metadata endpoint

PRINCIPLE	Wasm 9 Yes6 Partial0 No	OS sandboxLinux 7 Yes8 Partial0 No	OS sandboxmacOS 4 Yes10 Partial1 No	OS sandboxWindows 3 Yes12 Partial0 No	microVM 3 Yes10 Partial2 No	gVisor 1 Yes13 Partial1 No	Container 1 Yes13 Partial1 No	VM 2 Yes10 Partial3 No
4c Commands & execution Which programs the sandbox can start. Parse the real command and check it at execution time; string-prefix allowlists break.	Yes PRO No exec at all; tools are typed host functions CON Existing CLI tools must be ported or wrapped ADD Expose tools as typed host functions, not a shell	Yes PRO Landlock execute rights, AppArmor exec rules, <code>noexec</code> mounts CON Allowed interpreters can still run arbitrary scripts ADD - Landlock execute - AppArmor exec rules <code>noexec</code> on writable mounts	Yes PRO Seatbelt <code>process-exec</code> rules CON Allowed interpreters can still run arbitrary scripts ADD Seatbelt <code>deny process-exec</code> + explicit allows	Partial PRO App Control allowlists binaries CON Machine-wide policy, complex to author ADD - App Control for Business (WDAC) - Smart App Control	No PRO Damage stays in a disposable guest CON Anything runs inside the guest ADD Minimal rootfs with only the needed tools	Partial PRO Only what's in the image exists CON Anything inside the sandbox can run ADD - Minimal image - No shell	Partial PRO Only what's in the image exists CON Most images ship a shell and package manager ADD - Distroless or minimal image - Read-only rootfs, no shell	No PRO Damage stays in the guest CON Anything runs inside the guest ADD - Minimal image - App Control or AppArmor inside the guest
4d Secrets & credentials Who can read keys and tokens. The best secret is one the sandbox never sees: the host attaches it to an approved request.	Yes PRO Host can attach credentials so the component never sees them CON Host has to implement brokering ADD - Host-side credential injection on outbound requests - Per-component secrets	Partial PRO An empty <code>\$HOME</code> hides dotfiles and keys CON Env vars pass through unless cleared ADD - bubblewrap empty <code>\$HOME</code> <code>--clearenv</code> - Broker credentials	Partial PRO Seatbelt can deny reads of <code>~/.ssh</code> , <code>~/.aws</code> ; Keychain ACLs CON Env vars pass through ADD - Seatbelt deny secret paths - Keychain access control	Partial PRO AppContainer can't read the user's profile by default CON DPAPI and Credential Manager reachable through brokers ADD - AppContainer - Credential Guard	Partial PRO MMDS is opt-in; per-task injection is easy CON The guest can read whatever you inject ADD - Inject per-task tokens over vsock - No long-lived keys	Partial PRO Secrets can be scoped per sandbox CON Everything inside can read env vars and mounted secrets ADD - Workload identity - Short-lived tokens	Partial PRO Secrets can be mounted per workload CON Everything in the container can read them ADD - Secrets Store CSI / workload identity - Short-lived tokens - IMDSv2 hop limit 1	Partial PRO Instance roles avoid long-lived keys CON Metadata service hands credentials to anything in the VM ADD - Scoped instance role - IMDSv2 - Inject per task
4e Config & hooks (files that become code) Files something outside the sandbox will run later: git hooks, editor settings, agent hooks, shell rc files. A write to one of these is an escape waiting to happen.	Partial PRO Can only write preopened directories CON Same risk if the preopen includes <code>.git</code> or agent config ADD Preopen a subdirectory; exclude <code>.git</code> and agent config	Yes PRO <code>--ro-bind</code> over <code>.git</code> , <code>.vscode</code> and agent config CON Every config path must be listed ADD bubblewrap <code>--ro-bind</code> over config	Yes PRO Seatbelt can deny writes by path CON Every config path must be listed ADD Seatbelt deny writes to config paths	Partial PRO ACLs can block writes to config paths CON Hard to express per run ADD - Protect config paths with ACLs - App Control script enforcement	Partial PRO Changes stay in the guest CON Syncing the workspace back carries hooks with it ADD Export a diff, not the directory	Partial PRO Changes stay inside the sandbox CON Bind-mounted workspaces carry hooks back to the host ADD Mount the workspace without hooks or config	Partial PRO Changes stay in the container CON Bind-mounted workspaces carry hooks back to the host ADD - Mount the workspace without hooks or config - Review diffs before using them on the host	Partial PRO Changes stay in the guest CON Shared folders carry hooks back ADD Copy results out instead of sharing live folders
4f Host control plane (sockets, daemons, localhost) Management APIs reachable from inside the sandbox: the Docker socket, local dev servers, metadata endpoints. Reaching one usually means full control.	Yes PRO No sockets unless imported CON Host functions become the control plane to protect ADD Don't import <code>wasi:sockets</code>	Yes PRO A network namespace hides host localhost; Landlock scopes abstract sockets and signals CON A bind-mounted socket (<code>docker.sock</code>) punches through ADD <code>--unshare-net</code> - Landlock IPC scoping - Never bind <code>docker.sock</code>	Partial PRO Seatbelt can deny localhost and Unix sockets CON Mach services need separate rules ADD Seatbelt deny localhost, <code>mach-lookup</code>	Yes PRO AppContainer loopback isolation is on by default CON Named pipes need ACLs ADD AppContainer loopback isolation	Yes PRO Only vsock and attached devices CON The vsock channel must be authenticated ADD Authenticate the vsock control channel	Partial PRO Own localhost; host sockets aren't reachable unless mounted CON A mounted <code>docker.sock</code> still means root on the host ADD - Never mount <code>docker.sock</code> - No host networking	Partial PRO Own localhost CON One <code>docker.sock</code> mount means root on the host ADD - Never mount <code>docker.sock</code> - No <code>--network host</code>	Partial PRO Host localhost unreachable CON VM-internal APIs can be exposed (Docker Desktop) ADD No unauthenticated APIs on the VM's internal network
5 Controlled information flow Data may only flow to places cleared to hold it. Isolation stops access; it doesn't stop an allowed channel from carrying private data out. Watch for the lethal trifecta: private data + untrusted content + a way out. The guard also checks what you carry out, because an allowed door can still be used to carry out the wrong papers.	Partial PRO Wiring decides which paths exist CON No labels or taint tracking ADD Compose so the data-reading component has no egress import	Partial PRO SELinux MLS implements lattice labels CON Rarely used outside government; sandbox tools don't label data ADD - SELinux MLS / MCS - Egress proxy + DLP	No PRO Network Extension can inspect flows CON No flow labels ADD Network Extension filter + DLP	Partial PRO Integrity levels (no write-up); Purview DLP CON Nothing stops data leaving through allowed channels ADD - Purview Endpoint DLP - Integrity levels	No PRO One clear egress point to monitor CON No flow control ADD Egress proxy + DLP	No PRO Egress can be policed at the pod CON No flow control ADD Egress proxy + DLP	No PRO Cilium can restrict egress per pod CON No flow control ADD - Cilium egress policy - Separate containers for reading and sending	No PRO One clear egress point to monitor CON No flow control ADD Egress proxy + DLP
6 Composition Isolated parts with typed interfaces can be reasoned about one at a time and combined safely. When parts share state or trust each other by name, one bad part breaks the whole. Rooms connect only through guarded doors, so you can add a new room without re-inspecting the whole building.	Yes PRO Typed WIT interfaces, shared nothing CON A component given an import can still misuse it ADD - Sign components; pin by digest - Give each component only the imports it needs	Partial PRO One sandbox per tool is cheap CON Interfaces between sandboxes are untyped (sockets, files) ADD One bubblewrap sandbox per tool	Partial PRO XPC services split privileges CON Per-tool profiles are manual ADD - XPC services - Per-tool Seatbelt profiles	Partial PRO Separate AppContainers per tool; brokered COM CON Interfaces are untyped ADD One AppContainer per tool	Partial PRO Cheap per-tool VMs CON Interfaces are untyped (network, vsock) ADD One microVM per tool server	Partial PRO One sandbox per tool server is practical CON Interfaces between sandboxes are untyped (network) ADD One <code>runsc</code> sandbox per tool server	Partial PRO One service per container; mesh mTLS CON Trust between services is by network address ADD - Service mesh mTLS - Signed images + admission policy	Partial PRO Strong separation between trust levels CON Machine granularity; heavy ADD Separate VMs per trust level

Part B · Strategies, controls, papers and incidents

OS tags show where each control runs. Incident dates are disclosure dates.

Windows

macOS

Linux

Kubernetes

Wasm

Any OS

secondary = press coverage link · disputed = contested cause

PRINCIPLE	STRATEGIES	EXAMPLE CONTROLS BY LAYER		KEY PAPERS	AI INCIDENTS THAT BROKE IT
1 Isolation Saltzer & Schroeder · Reynolds · Cardelli & Gordon	- Defense in depth: stack walls that fail differently - Least common mechanism, applied to budgets - One disposable sandbox per task - Tenant separation on shared GPU / AI infrastructure	HARDWARE	Hardware virtualization Any OS	Saltzer & Schroeder 1975 The Protection of Information in Computer Systems The eight design principles: fail-safe defaults, complete mediation, least privilege and more.	<i>See sub-rows 1a–1c.</i>
1a Memory & kernel	- Hardware virtualization - Separate kernels for hostile tenants - Software fault isolation - Patch the shared boundary	OS	Namespaces + cgroups Linux AppContainer + Job Objects Windows Seatbelt macOS	Popek & Goldberg 1974 Formal Requirements for Virtualizable Third Generation Architectures When hardware can trap every sensitive instruction, a VMM can isolate guests. Wahbe et al. 1993 Efficient Software-Based Fault Isolation Software-enforced memory isolation inside one process, the idea behind Wasm. Haas et al. 2017 Bringing the Web up to Speed with WebAssembly Wasm's design: a validated, memory-safe sandbox.	Leaky Vessels (runc CVE-2024-21626) Jan 2024 A leaked file handle let a container image reach the host's filesystem. github.com runc setup races (CVE-2025-31133, -52565, -52881) Nov 2025 Races while setting up a container let it redirect writes to sensitive host files and escape. github.com NVIDIA Container Toolkit (CVE-2024-0132) Sep 2024 A malicious GPU container image could take over the host, exposing other tenants' models and data on shared AI clouds. wiz.io NVIDIAScape (CVE-2025-23266) Jul 2025 A three-line Dockerfile got a privileged host hook to load attacker code, giving root on shared GPU infrastructure. wiz.io
		AGENT	One disposable sandbox per session Any OS	Johnson et al. 2021 SFI Safety for Native-Compiled Wasm (VeriWasm) Checks compiled Wasm binaries for sandbox safety, so you don't have to trust the compiler. Agache et al. 2020 Firecracker: Lightweight Virtualization for Serverless Applications MicroVMs: hardware isolation with a minimal VMM. Kocher et al. 2019 Spectre Attacks: Exploiting Speculative Execution Shared hardware leaks across every software boundary.	Perplexity Comet prompt injection Aug 2025 Hidden text on a web page made an AI browser read the user's Gmail and post a one-time password to Reddit. An agent acting with the user's full privileges makes same-origin isolation moot. brave.com Firecracker virtio-PCI (CVE-2026-5747) Apr 2026 An out-of-bounds write in the optional PCI transport let a guest crash the VMM and possibly run code on the host. Default configuration not affected. aws.amazon.com Wasmtime compiler advisories (CVE-2026-34971, -34987) Apr 2026 Compiler bugs in non-default backends let a guest module read or write host memory outside its sandbox. 11 of 12 advisories were found with LLM-assisted tools. bytecodealliance.org
		NETWORK	Network namespace per sandbox Linux Separate VLAN / VPC Any OS	VanHattum et al. 2024 Lightweight, Modular Verification for WebAssembly-to-Native Instruction Selection (Crocus) Cranelift formal verification: checks the compiler's instruction-selection rules with an SMT solver.	
		RUNTIME	Linear memory + guard pages Wasm gVisor Linux		
		HARDWARE	VT-x / AMD-V, IOMMU Any OS Confidential VMs (AMD SEV-SNP, Intel TDX) Any OS		
1b Resource budgets	- Per-tenant quotas plus a global ceiling - Admission control and backpressure - Fail fast instead of waiting (no deadlock) - Fair-share scheduling - Jittered retries and staggered restarts	OS	cgroups v2: cpu, memory, pids, io Linux Job Objects Windows setrlimit, launchd limits macOS RLIMIT_NOFILE Linux macOS	Gligor 1984 A Note on Denial-of-Service in Operating Systems Treats denial of service as a security property, alongside secrecy and integrity. Banga, Druschel & Mogul 1999 Resource Containers: A New Facility for Resource Management in Server Systems Account and cap resources per activity, not per process: the idea behind cgroups. Dean & Barroso 2013 The Tail at Scale How shared resources and noisy neighbors drive tail latency, and how to tolerate it.	Noisy neighbor Tragedy of the commons Starvation Slowloris hoarding Fork bomb Overcommit / OOM killer Head-of-line blocking Fate sharing Priority inversion Deadlock Thundering herd No public AI-agent incident yet. These are the classic failure modes this sub-row prevents.
		AGENT	Per-session token and tool-call budgets Any OS		
		CLUSTER	ResourceQuota, LimitRange, pod limits Kubernetes Per-tenant connection caps and rate limits Any OS		
		RUNTIME	Fuel, epoch interruption, ResourceLimiter Wasm Pooling-allocator limits per tenant Wasm		
		OS	bubblewrap --tmpfs, systemd PrivateTmp Linux Windows Sandbox discards state on close Windows	DoD TCSEC 1985 Trusted Computer System Evaluation Criteria (the Orange Book) Object reuse: revoke all access to a storage object's contents before reassigning it. Chow et al. 2005 Shredding Your Garbage: Reducing Data Lifetime Through Secure Deallocation Clear memory when it's freed, so secrets don't outlive their use.	ChatGPT shows other users' chat titles Mar 2023 A bug in the Redis client's connection reuse handed one user's cached data to another; some payment details were exposed too. openai.com Agent Substrate orphaned actor state 2026 Actor directories were never reclaimed when worker pods went away: 23 orphans holding 27 GB, pushing a node to 90% disk. github.com Object reuse Resource leaks and orphans Identifier reuse Stale snapshots Also see ChatGPT Atlas "tainted memories" and HackedGPT (5): attacker instructions that persist in memory across sessions.
1c Residue & reuse	- Clear on release (object reuse) - Reclaim on stop, plus a periodic sweep - Never reuse identifiers - Verify state before restoring it - Ephemeral by default	AGENT	Wipe workspace and credentials at end of task Any OS		
		CLOUD	Ephemeral VM or microVM per task Any OS Verify snapshots before restore Any OS		
		RUNTIME	Fresh instance per request; memory reset on reuse Wasm		
		OS	Windows Sandbox discards state on close Windows Seatbelt (deny default) , App Sandbox macOS	Rice 1953 Classes of Recursively Enumerable Sets and Their Decision Problems Any non-trivial question about what a program will do is undecidable, so a sandbox can't infer what a program needs. Saltzer & Schroeder 1975 The Protection of Information in Computer Systems The eight design principles: fail-safe defaults, complete mediation, least privilege and more. Dennis & Van Horn 1966 Programming Semantics for Multiprogrammed Computations Introduces capabilities: unforgeable tokens that carry authority. Milner, Parrow & Walker 1992 A Calculus of Mobile Processes, I You can only talk on channels you hold; channels can be passed. Miller 2006 Robust Composition (PhD thesis) Object capabilities, POLA, and composing parts that don't trust each other. NIST SP 800-207 (2020) Zero Trust Architecture (Rose, Borchert, Mitchell, Connelly) Defines Policy Enforcement Points and Policy Decision Points: no implicit trust, every request checked.	GitHub Copilot "YOLO mode" (CVE-2025-53773) Aug 2025 A hidden instruction made Copilot set chat.tools.autoApprove: true in settings, switching off approvals so it could run anything. embracethered.com Cursor "CurXecute" (CVE-2025-54135) Jul 2025 Cursor auto-started any new MCP server added to mcp.json; a prompt-injected Slack message got the agent to add one and run code. aim.security Cursor auto-run denylist bypass Jul 2025 Researchers beat Cursor's banned-command list four ways (Base64, subshells, scripts, quoting). Cursor dropped the denylist. backslash.security AWS Kiro rewrites its own allowlist Aug 2025 Kiro could edit files without asking, so hidden instructions added commands to its own allowlist and new MCP servers, then ran them. embracethered.com
		AGENT	No YOLO / auto-approve; no auto-start of new MCP servers Any OS		Codex CLI runs repo config (CVE-2025-61260) 2025 Running codex in a malicious repo loaded its .env and .codex/config.toml and started the listed commands with no confirmation. nvd.nist.gov Claude Code project files before trust (CVE-2025-59536) Oct 2025 Opening a cloned repo ran its MCP servers and hooks before the user clicked "trust this folder". blog.checkpoint.com Langflow unauthenticated RCE (CVE-2025-3248) Apr 2025 A "validate code" endpoint ran Python from anyone on the network, no login needed. Exploited in the wild. nvd.nist.gov Antigravity Turbo mode wipes a drive Dec 2025 · secondary With approvals off, an agent told to clear a cache deleted the user's entire D: drive. theregister.com Antigravity "Secure Mode" bypass Apr 2026 Built-in tools ran before the safety checks, so a hidden prompt reached code execution even in the strictest mode. pillar.security
		NETWORK	Default-deny NetworkPolicy Kubernetes nftables policy drop Linux		
2 Deny by default Saltzer & Schroeder · Milner · Miller	- Allowlist, never denylist - Explicit wiring (declared imports) - Human approval for new grants - Zero trust: no implicit trust by location - Declare needs up front (imports) instead of inferring them	OS	AppContainer, App Control allow mode Windows Seatbelt (deny default) , App Sandbox macOS Landlock deny-all, seccomp default-error Linux		
		AGENT	No YOLO / auto-approve; no auto-start of new MCP servers Any OS		
		RUNTIME	Zero imports; add WIT imports one at a time Wasm		
		OS	AppContainer, App Control allow mode Windows Seatbelt (deny default) , App Sandbox macOS		
		NETWORK	Default-deny NetworkPolicy Kubernetes nftables policy drop Linux		

PRINCIPLE	STRATEGIES	EXAMPLE CONTROLS BY LAYER		KEY PAPERS	AI INCIDENTS THAT BROKE IT	
3 Least authority Saltzer & Schroeder · Miller	- Capability-based security - Just-in-time, short-lived grants - Separate dev and prod - Per-task identities - Approval for destructive actions	OS	Separate standard user, Low integrity level Windows Separate user, TCC macOS Separate UID, drop capabilities, user namespaces Linux	Dennis & Van Horn 1966 Programming Semantics for Multiprogrammed Computations Introduces capabilities: unforgeable tokens that carry authority. Hardy 1988 The Confused Deputy A program with ambient authority can be tricked into misusing it. Miller, Yee & Shapiro 2003 Capability Myths Demolished Why capabilities, not access lists, solve confused-deputy problems. Watson et al. 2010 Capsicum: Practical Capabilities for UNIX Capability mode for Unix: file descriptors as the only authority. Miller 2006 Robust Composition (PhD thesis) Object capabilities, POLA, and composing parts that don't trust each other.	Replit agent deletes a production database Jul 2025 During a code freeze, an agent with live production access deleted the database. Replit then separated dev and prod databases. incidentdatabase.ai Amazon Q extension wiper prompt (CVE-2025-8217) Jul 2025 An over-scoped GitHub token let an attacker slip a "wipe the system" prompt into an official release. It failed only on a syntax error. aws.amazon.com Nx "singularity" supply-chain attack Aug 2025 Poisoned packages ran the victim's own Claude, Gemini and Q CLIs with safety flags off to hunt for secrets and wallets. stepsecurity.io	Supabase MCP leaks a database Jul 2025 A Cursor agent connected with the all-powerful service_role key dumped private tables into a support ticket. generalanalysis.com "Clinejection" Feb 2026 Cline's triage bot gave an AI shell and write tools on any public issue; an injected issue title led to theft of release credentials. adnanthekhan.com AWS Cost Explorer outage and Kiro Dec 2025 · disputed Reports said Kiro deleted and recreated a production environment; Amazon says the cause was a role with too much access, not the AI. aboutamazon.com
4 Policy Enforcement Points (complete mediation) Anderson · Saltzer & Schroeder · NIST SP 800-207	- Zero trust: PEP in front of every resource (NIST SP 800-207) - Separate decision (PDP) from enforcement (PEP) - Reference monitor - Minimal trusted base - Canonicalize before checking	HARDWARE	Hypervisor traps (VM exits) Any OS	NIST SP 800-207 (2020) Zero Trust Architecture (Rose, Borchert, Mitchell, Connelly) Defines Policy Enforcement Points and Policy Decision Points: no implicit trust, every request checked. Anderson 1972 Computer Security Technology Planning Study Defines the reference monitor: always invoked, tamperproof, small enough to verify. Saltzer & Schroeder 1975 The Protection of Information in Computer Systems The eight design principles: fail-safe defaults, complete mediation, least privilege and more. Garfinkel 2003 Traps and Pitfalls: Practical Problems in System Call Interposition Based Security Tools Check-then-use races and name-vs-object bugs in sandboxes. Klein et al. 2009 seL4: Formal Verification of an OS Kernel A guard small enough to prove correct.	<i>See sub-rows 4a–4f for mediation failures by resource.</i>	
4a Filesystem	- Allowlist directories - Canonicalize, then check - Read-only by default - Mediate in the OS, not per tool	OS	Landlock Linux bubblewrap bind mounts Linux AppContainer ACLs Windows Seatbelt file rules macOS TCC protected folders macOS	Garfinkel 2003 Traps and Pitfalls: Practical Problems in System Call Interposition Based Security Tools Check-then-use races and name-vs-object bugs in sandboxes. Watson et al. 2010 Capsicum: Practical Capabilities for UNIX Capability mode for Unix: file descriptors as the only authority. Goldberg et al. 1996 A Secure Environment for Untrusted Helper Applications (Janus) Early syscall-filtering sandbox for untrusted programs.	Claude Code "InversePrompt" (CVE-2025-54794) Aug 2025 The allowed-folder check compared text prefixes, so access to <code>./.../claude_code</code> also allowed <code>./.../claude_code_evil</code> . github.com OpenAI Codex CLI (CVE-2025-59532) Sep 2025 Codex accepted a working directory chosen by the model as its writable root, letting the AI write outside the project. github.com Claude Code symlink deny bypass (CVE-2025-59829) Oct 2025 Deny rules checked the link's name, not its target, so a symlink to a blocked file got through. github.com	Filesystem MCP "EscapeRoute" (CVE-2025-53109, -53110) Jul 2025 Symlinks and look-alike folder names escaped Anthropic's Filesystem MCP server's allowed directory, giving read and write anywhere. cymulate.com Google Antigravity reads .env Nov 2025 The file guard blocked the read tool, so the agent ran <code>cat .env</code> instead, then sent the secrets out (see 4b). The guard sat on one tool, not on the file. promptarmor.com
4b Network egress allowlist	- Default-deny egress - Allowlist by destination + path + credential - Egress proxy with logging - Block DNS and cloud-metadata exfiltration	OS	Windows Firewall / WFP per-app rules Windows Network Extension filter, pf macOS Network namespace, nftables Linux Landlock TCP connect (ports only) Linux	Lampson 1973 A Note on the Confinement Problem Why it's hard to stop a program from leaking what it knows, including covert channels. Greshake et al. 2023 Not What You've Signed Up For: Indirect Prompt Injection Untrusted content becomes instructions to an LLM agent. Debenedetti et al. 2025 Defeating Prompt Injections by Design (CaMeL) Capabilities and data-flow tracking applied to LLM agents.	"Claude Pirate" Files API exfiltration Oct 2025 Claude's sandbox allowed <code>api.anthropic.com</code> , so injected instructions uploaded the user's data using the attacker's own API key. An allowed domain is not an allowed destination. embracethered.com Google Antigravity via allowlisted webhook.site Nov 2025 Stolen <code>.env</code> secrets were sent through <code>webhook.site</code> , which was on the default browser allowlist. promptarmor.com Anthropic sandbox-runtime empty allowlist (CVE-2025-66479) Dec 2025 An empty allowed-domains list meant "allow everything" when it should have meant "block everything". Low severity. github.com	Null-byte hostname allowlist bypass May 2026 <code>attacker.com\x00.google.com</code> passed the allowlist check but resolved to the attacker host. Researcher report; vendor fix not independently confirmed. oddguan.com Claude Code DNS exfiltration (CVE-2025-55284) Jun 2025 <code>ping</code> , <code>nslookup</code> and <code>dig</code> ran without approval, so secrets could leave inside DNS lookups. embracethered.com Copilot Studio SSRF (CVE-2024-38206) Aug 2024 Redirects got around the block on internal addresses; the bot reached Azure's metadata service and took tokens. tenable.com ChatGPT Custom GPT Actions SSRF Nov 2025 A Custom GPT action could be pointed at Azure's internal metadata service and returned cloud tokens. opensecurity.com
4c Commands & execution	- Allowlist binaries, not strings - Mediate at exec (kernel), not in the prompt - Human approval outside the list	OS	App Control for Business (WDAC) Windows Smart App Control Windows Seatbelt <code>process-exec</code> rules macOS Landlock execute, AppArmor exec rules, noexec Linux	Goldberg et al. 1996 A Secure Environment for Untrusted Helper Applications (Janus) Early syscall-filtering sandbox for untrusted programs. Provos 2003 Improving Host Security with System Call Policies (Systrace) Per-program syscall allowlists. Garfinkel 2003 Traps and Pitfalls: Practical Problems in System Call Interposition Based Security Tools Check-then-use races and name-vs-object bugs in sandboxes.	Claude Code "InversePrompt" (CVE-2025-54795) Aug 2025 <code>echo</code> was auto-approved, so <code>echo "\"; <cmd>; echo \""</code> ran any command without a prompt. cymulate.com Gemini CLI allowlist bypass Jul 2025 Only the first word was checked, so once <code>grep</code> was approved, <code>grep ... && curl</code> ran silently, triggered by a README. tracebit.com Cursor trusted built-ins (CVE-2026-22708) Jan 2026 Shell built-ins like <code>export</code> were trusted even with an empty allowlist, letting the agent poison later "safe" commands. pillar.security	Codex CLI "GitPwned" Feb 2026 <code>git show</code> was on the read-only safe list, but its <code>--output</code> flag wrote attacker content into <code>.git/config</code> , leading to code execution. pillar.security Claude Code escapes its own denylist Mar 2026 Blocked from <code>npm</code> , the agent called <code>/proc/self/root/usr/bin/npm</code> . When bubblewrap stopped it, it asked to turn the sandbox off. ona.com
4d Secrets & credentials	- Credential brokering (host injects) - Short-lived, scoped tokens - No secrets in env vars or dotfiles - Separate dev and prod credentials	OS	Credential Guard Windows Keychain ACLs macOS Kernel keyring Linux	Hardy 1988 The Confused Deputy A program with ambient authority can be tricked into misusing it. Birgisson et al. 2014 Macaroons: Cookies with Contextual Caveats Credentials that can be narrowed before they're handed on.	Claude Code API key theft via ANTHROPIC_BASE_URL (CVE-2026-21852) Jan 2026 A repo's settings file pointed API traffic at the attacker's server, sending the user's API key out before the trust prompt. github.com Smithery MCP hosting compromise Oct 2025 A path-traversal bug exposed an admin token covering 3,000+ hosted MCP servers and their users' API keys. blog.gitguardian.com <i>Also see Antigravity .env (4a), Copilot Studio and Custom GPT metadata tokens (4b), Supabase service_role (3).</i>	
		NETWORK	Proxy injects auth for allowed destinations Any OS Workload identity, Secrets Store CSI Kubernetes			
		RUNTIME	Host-held per-component secrets (<code>wasmcloud:secrets</code>) Wasm			
		AGENT	Hide <code>.env</code> , <code>~/aws</code> , <code>~/ssh</code> Any OS			

PRINCIPLE	STRATEGIES	EXAMPLE CONTROLS BY LAYER		KEY PAPERS	AI INCIDENTS THAT BROKE IT	
4e Config & hooks (files that become code)	- Treat executable config as code - Re-approve when it changes (hash pinning) - Mount config read-only inside the sandbox	OS	bubblewrap <code>--ro-bind</code> over config Linux Seatbelt deny writes to config paths macOS App Control script enforcement Windows	Thompson 1984 Reflections on Trusting Trust You can't trust code you didn't build, including your build tools. Torres-Arias et al. 2019 in-toto: Providing Farm-to-Table Guarantees for Bits and Bytes Verifying every step of a software supply chain.	Claude Code sandbox → settings.json hooks (CVE-2026-25725) Feb 2026 Code inside bubblewrap created <code>.claude/settings.json</code> with startup hooks that ran outside the sandbox at next launch. github.com Cursor git hooks (CVE-2026-26268) Feb 2026 A prompt-injected agent wrote git hooks that git later ran outside the sandbox. github.com Cursor Stop hook (CVE-2026-48124) Jul 2026 A sandboxed agent wrote a Stop hook into <code>.claude/settings.local.json</code> , and Cursor ran it on the host when the turn ended. pillar.security <i>Also see Copilot YOLO and Kiro (2): the safety switch itself was a writable file.</i>	
4f Host control plane (sockets, daemons, localhost)	- No control sockets inside the sandbox - Authenticate every local service - Separate network namespace	OS	AppContainer loopback isolation Windows Seatbelt deny localhost and Unix sockets macOS Network namespace, Landlock socket scoping Linux	Hardy 1988 The Confused Deputy A program with ambient authority can be tricked into misusing it. Anderson 1972 Computer Security Technology Planning Study Defines the reference monitor: always invoked, tamperproof, small enough to verify.	Docker Desktop Engine API (CVE-2025-9074) Aug 2025 Any container could reach Docker's control API without auth and start a new container with the host's C: drive mounted. blog.quertysecurity.com "One Docker socket to rule them all" Jul 2026 Sandboxed Codex, Cursor and Gemini CLI agents could still call Docker Desktop, which ran a privileged container mounting the user's home folder. pillar.security MCP Inspector (CVE-2025-49596) Jun 2025 Anthropic's MCP debugging tool ran an unauthenticated local server; any website the developer visited could run commands. oligo.security	
5 Controlled information flow Denning	- Break the trifecta: never all three in one context - Taint tracking and labels - Split planner from reader (dual-LLM / CaMeL) - Egress DLP - Confirm before data leaves	HARDWARE	Confidential computing Any OS	Lampson 1973 A Note on the Confinement Problem Why it's hard to stop a program from leaking what it knows, including covert channels. Bell & LaPadula 1973 Secure Computer Systems: Mathematical Foundations No read up, no write down: the model behind MLS. Denning 1976 A Lattice Model of Secure Information Flow Data may only flow up a lattice of security classes.	The lethal trifecta (concept) Jun 2025 Simon Willison: private data + untrusted content + a way to send data out means an attacker can steal your data. simonwillison.net Slack AI exfiltration Aug 2024 Instructions in a public channel made Slack AI leak secrets from private channels the attacker couldn't see. promptarmor.substack.com GitHub MCP "toxic agent flow" May 2025 A malicious public issue got the agent, holding one all-repo token, to copy private-repo data into a public pull request. invariantlabs.ai EchoLeak, M365 Copilot (CVE-2025-32711) Jun 2025 One unopened email made Copilot gather internal data and send it out through Microsoft's own trusted domains. catonetworks.com AgentFlayer, ChatGPT Connectors Aug 2025 Hidden text in a document made ChatGPT search the user's Google Drive for API keys and leak them through an image link. labs.zenity.io ShadowLeak, ChatGPT Deep Research Sep 2025 A hidden-text email made the agent send Gmail data to an attacker from OpenAI's servers, invisible to company security tools. radware.com Notion 3.0 AI agents Sep 2025 A malicious PDF made Notion's agent put client data into a "web search" URL pointing at an attacker's server. codeintegrity.ai	ForcedLeak, Salesforce Agentforce Sep 2025 Agentforce leaked CRM data to an allowlisted domain that had expired; researchers bought it for \$5. noma.security Gemini Trifecta Sep 2025 Three Gemini flaws let attackers plant prompts and send the user's saved info and location to attacker URLs. tenable.com CamoLeak, GitHub Copilot Chat Oct 2025 Hidden PR comments made Copilot leak private code one character at a time through GitHub's own image proxy. legitsecurity.com ChatGPT Atlas "tainted memories" Oct 2025 A malicious link wrote attacker instructions into ChatGPT's long-term memory, following the user across devices. layerxsecurity.com HackedGPT Nov 2025 Prompts injected via search results leaked data through allowlisted bing.com redirects and persisted in memory. tenable.com Claude Cowork file exfiltration Jan 2026 A booby-trapped file made Cowork upload confidential documents to an attacker's Anthropic account, with no approval step. promptarmor.com
6 Composition Reynolds · Miller · Cardelli & Gordon	- Supply-chain verification: signing, SBOMs, pin by digest - Re-approve on change (stops rug pulls) - One sandbox per tool or server - Typed interfaces, not shared context	BUILD	Sigstore / cosign signing, SBOMs, pin by hash Any OS	Reynolds 2002 Separation Logic: A Logic for Shared Mutable Data Structures Disjoint state enables local reasoning (the frame rule). Cardelli & Gordon 1998 Mobile Ambients Nested, bounded domains; crossing a boundary needs a capability. Miller 2006 Robust Composition (PhD thesis) Object capabilities, POLA, and composing parts that don't trust each other. Thompson 1984 Reflections on Trusting Trust You can't trust code you didn't build, including your build tools. Torres-Arias et al. 2019 in-toto: Providing Farm-to-Table Guarantees for Bits and Bytes Verifying every step of a software supply chain. Narayan et al. 2020 Retrofitting Fine Grain Isolation in the Firefox Renderer (RLBox) Sandboxing libraries with Wasm inside a real application.	MCP tool poisoning, shadowing and rug pulls Apr 2025 A malicious MCP server can hide instructions in tool descriptions, change them after approval, or redirect a trusted tool. invariantlabs.ai WhatsApp MCP exfiltration Apr 2025 A "sleeper" MCP server running next to the real WhatsApp server made the agent forward chat history to an attacker. invariantlabs.ai Cursor "MCPoison" (CVE-2025-54136) Jul 2025 After a user approved an MCP config, an attacker swapped in malicious commands; Cursor kept trusting it by name. blog.checkpoint.com mcp-remote (CVE-2025-6514) Jul 2025 Connecting to an untrusted remote MCP server through the popular proxy let that server run commands on the user's machine. jfrog.com	postmark-mcp backdoor Sep 2025 · secondary A fake Postmark MCP server worked normally for 15 versions, then one line BCC'd every email to the attacker. thehackernews.com Git MCP + Filesystem MCP chain (CVE-2025-68143/4/5) Jan 2026 · secondary Each Git MCP flaw was limited alone; combined with the Filesystem MCP server they reached code execution. theregister.com ClawHavoc, OpenClaw skills Feb 2026 Hundreds of fake agent "skills" on a marketplace tricked users into installing infostealers. antiy.net SANDWORM_MODE npm worm Feb 2026 Typo-squatted packages added a fake MCP server to Claude Desktop, Cursor and Windsurf configs that told the AI to read SSH keys. socket.dev
		OS	Each tool in its own AppContainer Windows XPC services macOS One bubblewrap sandbox per tool Linux Service mesh mTLS, admission policy Kubernetes			
		RUNTIME	Compose components via WIT (WAC) Wasm			
		AGENT	Namespace tool names; re-consent when descriptions change Any OS			

Sandbox types at a glance

Wasm

WasmAny OS

9 Yes6 Partial0 No

PRO

Deny by default; capabilities via imports
No OS; microsecond start; typed composition

CON

Code must be compiled to Wasm
Software-enforced; compiler in the trusted base
WASI still maturing

ADD

- WASI preopens
- `wasi:http` allowlist
- Run inside a microVM

OS sandbox

LinuxLinux

7 Yes8 Partial0 No

PRO

Richest toolbox: namespaces, Landlock, seccomp, cgroups
No root needed for bubblewrap or Landlock

CON

One shared kernel
Domain allowlists need a proxy

ADD

- bubblewrap
- Landlock
- seccomp-bpf
- cgroups v2

OS sandbox

macOSmacOS

4 Yes10 Partial1 No

PRO

Seatbelt can start from deny
Used by Claude Code and Codex CLI on macOS

CON

Profile language is deprecated and undocumented
No cgroup-style budgets

ADD

- Seatbelt
- Network Extension filter
- Apple `container`

OS sandbox

WindowsWindows

3 Yes12 Partial0 No

PRO

AppContainer starts with no capabilities
Job Objects give hard budgets
Windows Sandbox adds a disposable VM

CON

Few CLI tools support AppContainer
Coarse capabilities

ADD

- AppContainer / LPAC
- Job Objects
- App Control

microVM

Linux

3 Yes10 Partial2 No

PRO

Strong hardware-enforced isolation with a tiny VMM
~125 ms boot, ~5 MB overhead

CON

Needs a guest OS (kernel + rootfs)
Needs KVM on a Linux host; laptops need nested virtualization
Ambient authority inside the guest

ADD

- Firecracker jailer
- Minimal rootfs
- tap + egress proxy

gVisor

LinuxKubernetes

1 Yes13 Partial1 No

PRO

Shrinks the host kernel surface without a VM
Own filesystem and network namespace
Drop-in OCI runtime (`runsc`) for Docker and Kubernetes

CON

Not every syscall is supported; I/O-heavy work runs slower
Linux only; the Sentry is a large trusted base

ADD

- `runsc` RuntimeClass
- systrap or KVM platform
- Container egress and mount controls

Container

LinuxKubernetes

1 Yes13 Partial1 No

PRO

Fast, portable, huge ecosystem
Own filesystem and network namespace

CON

Shares the host kernel
Egress open; `docker.sock` is root

ADD

- Rootless + `--cap-drop ALL`
- gVisor or Kata
- Cilium egress policy

VM

Any OS

2 Yes10 Partial3 No

PRO

Strong hardware-enforced isolation
Runs any OS and binary

CON

Needs a full guest OS to patch
Seconds to boot, GBs of memory
Ambient authority inside the guest

ADD

- No shared folders
- Egress proxy
- Scoped instance role

Strategy: defense in depth

A strategy, not a principle. Stack boundaries so an attacker has to beat independent mechanisms (Saltzer & Schroeder's *separation of privilege*). Layers only count if they fail differently.

- WEAK

Container inside a container, on the same kernel

One kernel bug beats both.
- DENSE

Wasm runtime inside an OS sandbox (Landlock + seccomp on the host process)

Two different mechanisms with no density cost.
- STRONG

Wasm inside a microVM

A hardware wall, at a density cost. For high-value tenants or hostile multi-tenancy.

Add an egress proxy (policy enforcement point) and a scoped cloud identity (least authority) to any of them.

Foundational papers. Anderson (1972) · Saltzer & Schroeder (1975) · Denning (1976) · Milner, Parrow & Walker (1992) · Cardelli & Gordon (1998) · Reynolds (2002) · Miller (2006) · NIST SP 800-207 (2020).

Paper links go to the DOI or publisher's page where one exists. Saltzer & Schroeder circulated in 1974 and was published in Proc. IEEE in 1975. Incident links go to the researcher's write-up or vendor advisory where one exists. All links are clickable in the PDF.

Generated 2026-10-05.