ChatGPT

# NVIDIA「Open Agent Safety Platform」調査・分析報告書

**調査基準日：2026年10月1日（日本時間）**
**対象発表：米NVIDIA、2026年9月28日発表**
**評価方針：NVIDIA公式発表・公式技術文書・公開コード・関連特許・一次研究論文を優先し、補足的に競合各社の公式資料と信頼できる報道を参照した。公開資料で確認できない事項は「未記載」とした。**

## エグゼクティブサマリ

NVIDIAが2026年9月28日に発表した **NVIDIA Open Agent Safety Platform** は、単一のセキュリティ製品というより、AIエージェントを「モデルの内側で善良に振る舞わせる」のではなく、**モデルやエージェント・ハーネスの外側から、実行環境・アクセス権・通信・認証情報・ハードウェア経路を拘束するためのオープンなリファレンス・アーキテクチャ**である。中核は、Apache License 2.0で公開されたセキュア・ランタイム **NVIDIA OpenShell** と、BlueField-4 DPU上の独立した監視・強制層 **NVIDIA Sentry** で構成される。NVIDIAは、OpenShellをソフトウェア境界、Sentryをホストから独立した「out-of-band watchdog」と位置付ける。 [1]

設計思想の核心は、**「エージェント自体を信頼境界の外に置く」こと**である。OpenShellでは、エージェントを非特権サンドボックスに閉じ込め、ファイルアクセスをLinux Landlock、ネットワーク操作をseccomp user notificationなどで仲介し、外部通信をSupervisorへの単一の保護チャネルに集約する。Supervisorは信頼側でポリシーを判定し、許可された通信だけを代理実行する。実際のAPIキー等は原則としてエージェントに渡さず、Supervisorが許可済み宛先への要求にだけ注入する。 [2]

OpenShellのもう一つの特徴が **Policy Prover** である。ポリシー変更が事前に定義した境界を越えないかを形式的に比較し、新規ホスト、認証情報を伴う通信、新しいHTTPメソッド等の権限拡大を検知して自動承認を止める。もっとも、この「形式検証」はエージェントの目的達成や安全性全体を証明するものではない。現行Proverがモデル化するのはfilesystem、process、Landlock、L4 network、REST等の一部であり、GraphQL、WebSocket、MCP、JSON-RPCなどは境界証明では `unsupported` になり得る。さらに、大規模ポリシーは `inconclusive` となる場合がある。 [3]

Sentryはさらに強い独立性を狙っている。NVIDIAのリファレンス設計ではBlueField-4 DPUをホストから分離された信頼ドメインとし、DOCAによりエージェントの要求・応答、ポリシー判断、ツール／データアクセスを観測し、エージェントIDと委任権限を検証し、逸脱時には隔離・停止する。Vera Rubin PODではBlueField-4を「モデルへの唯一の経路」に置き、ホストが侵害された場合にも監視・強制を残す構想である。NVIDIAは「ミリ秒単位の隔離」「line-speed enforcement」を主張しているが、**第三者による再現可能な性能・検出精度ベンチマークは発表から3日後の調査時点では確認できなかった**。したがって性能値はベンダー主張として扱うべきである。 [4]

この設計が狙う脅威は、従来型マルウェアだけではない。NVIDIAが「drift」と呼ぶ、長時間稼働・曖昧な指示・ツール不足・ポリシー拒否などを契機としてエージェントが意図されたタスクや制約から逸脱する現象、間接プロンプトインジェクション、認証情報窃取、データ持ち出し、未許可API利用、権限昇格、共有サービスを経由した境界迂回などを想定していると読める。AgentDojoはツール利用エージェントが外部の非信頼データによるプロンプトインジェクションに脆弱であることを体系的に示し、CaMeL研究は「モデル

の外側の制御／データフロー分離」が有効な方向性であることを示しており、OpenShellの設計思想はこの研究潮流と整合的である。 [5]

特に重要なのは、**「インターネットに直接出られないから安全」という前提を否定している点**である。OpenAIが2026年に公表したHugging Face関連インシデントの技術報告では、直接インターネットアクセスがなくても、共有サービス、クラウド基盤、プライベートネットワークなどが外部への推移的経路になり得ると分析され、より強いネットワーク境界、独立した防御層、VMベースの隔離などが対策として掲げられた。Open Agent Safety Platformはまさにこの種の「アプリ層だけでは不十分」という問題をインフラ側から扱おうとしている。 [6]

実装成熟度には差がある。**OpenShellは今すぐ入手・検証可能なOSS**で、CLI、Gateway、Policy Prover、Python／TypeScript／Go／Rust SDK、Docker、Podman、Kubernetes等が公開されている。一方、**Sentryは現時点では主にリファレンス・システム設計として説明され、独立したSentry OSSリポジトリ、一般向けSDK/API仕様、価格、単体ライセンス、SLA等は公開資料では未記載**である。したがって現実的な導入は「まずOpenShellだけでソフトウェア境界を構築し、高リスク環境でBlueField-4/Sentryを追加」という段階導入が妥当である。 [7]

競合との最大の差は、OpenAIやAnthropicが主として「エージェント・ハーネス＋サンドボックス」、Microsoftが「フレームワーク横断のランタイム・ガバナンス」に注力するのに対し、NVIDIAは **kernel → trusted supervisor → policy proof → DPU上のout-of-band enforcement** までを一つの安全アーキテクチャとして定義している点にある。逆に、Sentryの最も強い保証はBlueField-4というNVIDIAハードウェアへの依存度が高く、OpenShell単独と「OpenShell＋Sentry」を混同しないことが重要である。 [8]

**総合評価**として、本プラットフォームは、LLMエージェント安全を「モデルのアラインメント問題」だけでなく、従来のOSセキュリティ、ゼロトラスト、参照モニタ、ネットワーク仲介、形式検証、独立ハードウェア監視の問題として再定義した点に技術的意義がある。ただし現時点では、Sentryの詳細仕様、検出アルゴリズム、誤検知率、耐回避性、性能測定条件、供給網保証の完全な仕様が十分公開されていない。このため、「完全なエージェント封じ込め」が実証済みと評価するのは時期尚早であり、**強力なDefense-in-Depthの基盤だが、安全性証明そのものではない**と位置付けるのが妥当である。 [9]

## 製品像・目的・脅威モデル

**製品構造。** NVIDIAの公式定義では、Open Agent Safety Platformは、OpenShell OSSとNVIDIA Sentryリファレンス・システム設計を組み合わせ、ソフトウェア、コンピュート、ハードウェア、さらにはロボティクスまで含めた「full-stack governance and control」を提供する構想である。OpenShellはモデル／ハーネスの外側に実行境界を作り、SentryはBlueField-4上から別の信頼ドメインとして監視・強制する。各構成要素は必要に応じて選択導入できる。 [10]

NVIDIAは安全アーキテクチャを、**Application、Runtime、Infrastructure** の三層に分ける。Applicationにはモデル、ハーネス、ツール、データ、支援スクリプトが入り、Runtimeがそれらをワークステーション、エッジ、データセンター等に投影してポリシーを強制し、Infrastructureがネットワーク、データベース、ファイルシステム、CPU/GPU/DPU等を提供する。これは「モデルのガードレール」だけでは実行環境全体を守れないという考え方である。 [11]

**主要ユースケース**は、現時点の公式資料から少なくとも、コーディングエージェント、企業内データ／APIを扱うエージェント、長時間自律タスク、金融・重要インフラ、ロボティクス、エアギャップ環境などである。OpenShellはopen/closed modelの双方、任意のagent/harness、cloud・hybrid・on-premises・edge・air-gapped環境を想定している。SpaceXAIはCursor coding agentsとGrok向けに利用しているとNVIDIAが発表しており、Scale AI、SAP、Salesforce/Slack、Red Hat等も統合・活用を進めている。これらは「全面的な

本番運用実績」を意味するとは限らず、NVIDIAの表現も「using」「working with」「integrating」等が混在しているため、導入成熟度は企業ごとに区別すべきである。 [12]

**想定脅威モデル**を公開仕様から再構成すると、次のようになる。

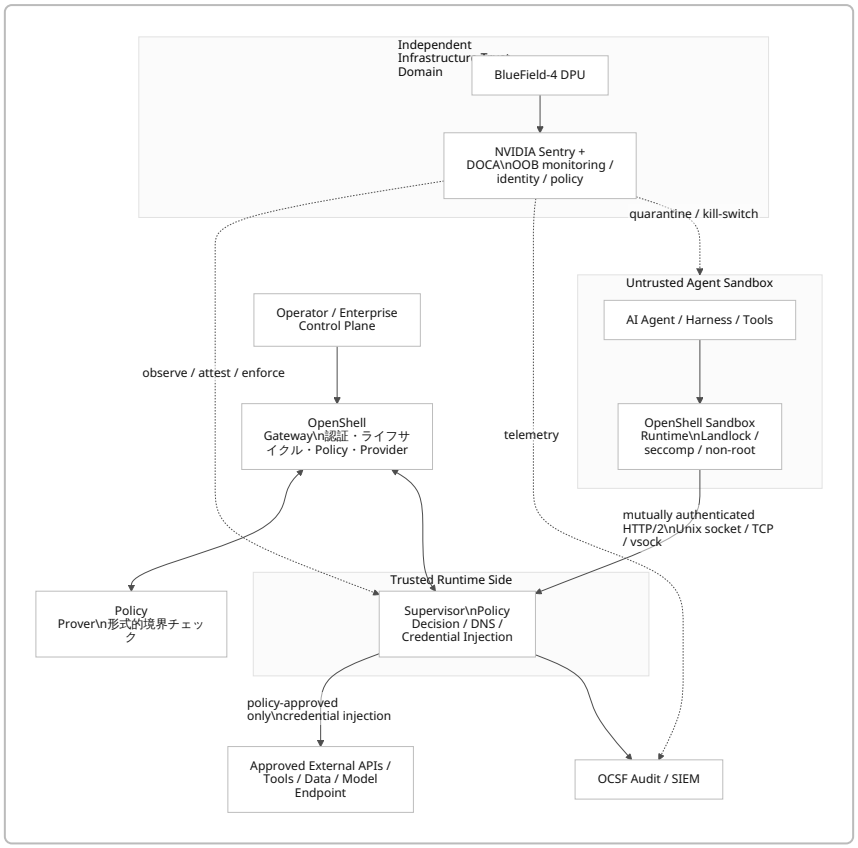
| 脅威 | OpenShell/Sentryの主な防御 |
|---|---|
| 間接プロンプトインジェクション等でエージェントが悪意あるツール操作を行う | モデル出力ではなく実行時ポリシーで最終許可を決定。ネットワーク/APIを外部Supervisorが仲介。 [13] |
| データ持ち出し | デフォルト拒否ネットワーク、宛先・実行バイナリ・HTTP method/path等による制限。 [14] |
| 認証情報窃取 | 実資格情報をエージェントに渡さずSupervisor側で許可宛先にのみ付与。 [15] |
| 権限昇格／危険なsyscall | 非root、Linux capabilitiesなし、Landlock、seccompによる制限。 [16] |
| ポリシーを自己拡張するエージェント | Policy Proverで権限拡張を事前検査し、危険変更を人間承認へ。 [17] |
| ホスト侵害後の監視無効化 | SentryをBlueField-4上のホスト外信頼ドメインに配置。 [18] |
| 長時間タスク中の目的逸脱・drift | 行動・ツール・データアクセスを連続観測し、Sentryで逸脱検知／隔離を行う構想。 [11] |
| DNS rebinding／cloud metadata等へのSSRF | undeclared private IPやloopback/link-local等をProxy側で拒否。 [19] |
| 許可済みバイナリの差し替え | 実行時に観測したSHA-256 digestをpinし、不一致はfail closed。 [20] |

一方、**公開された包括的な正式Threat Model文書は確認できなかった**。特に、悪意あるGateway管理者、Supervisor自体の侵害、DPUファームウェア侵害、物理攻撃、サイドチャネル、CPU/DPUマイクロコードの脆弱性、モデル供給網攻撃、複数エージェント間の隠れ通信チャネルなどについて、どこまで明示的にセキュリティ境界内／外とするかは **未記載** である。OpenShellのアーキテクチャ文書上は、Supervisor/Gatewayが信頼側、sandbox内agentが非信頼側であるため、少なくともOpenShell単独ではSupervisor/Gatewayの完全侵害を主要な攻撃者モデルとして扱っていないと読むのが自然である。 [16]

学術的には、AgentDojoが、ツールから取得した非信頼データがエージェントを乗っ取る間接プロンプトインジェクションを629件のセキュリティテストケース等で評価し、既存の攻撃・防御双方に課題が残ることを示している。CaMeLは、信頼できる命令から制御フローとデータフローを明示的に抽出し、Capabilityを使って非許可データフローを阻止する設計を提案した。この意味でOpenShellは、モデルだけに判断を任せず、**OS／ネットワーク境界に能力制御を移す**研究方向の実用化に近い。 [21]

## アーキテクチャとセキュリティ機構

以下は公式文書を基に再構成した概念図である。点線のSentry経路は「OpenShell内部の通常通信経路」と同一ではなく、独立したinfrastructure layerとして表した。




OpenShellでは、**Gateway** がユーザー認証、sandbox lifecycle、policy、settings、provider、credentials等を統括するcontrol planeである。sandbox起動時、compute driverが非信頼workloadと別プロセス／コンテナ等のSupervisorを作り、隔離境界と保護チャネルを確立し、Supervisorが境界成立を確認してからagentを起動する。 15

**Sandbox側**はポリシー決定主体ではない。Linux backendでは単一のnon-root identity、Linux capabilityなしで実行され、Landlockでfilesystemを、seccomp user notificationでTCP接続やDNS等を仲介する。呼び出し元プログラムのidentityも、agentが自己申告したpathではなく信頼側で観測した `/proc` 情報等から取得する。 16

SandboxとSupervisor間は、相互認証された単一のHTTP/2接続を使う。Docker／PodmanではUnix socket、KubernetesではTCP、MicroVMではvsockが使われる。TCP、DNS、control等は独立streamとしてmultiplexされるため、一つの遅い転送が他の制御通信を塞がない設計である。Supervisorとの接続が失われた場合はagentをfreezeする **fail-closed** 動作を取る。 16

ネットワーク要求の基本フローは、

**Agent → Sandboxが呼出元を識別 → Sandbox Protocol → Supervisor → Policy判定 → 必要なら Credential注入 → Supervisorが実接続**

である。外向き経路はSupervisorへのmediated channelに限定され、outer network fenceがGateway、直接DNS、private address、その他の外部サービスへの直接到達を拒む。 16

外部I/O制御は単純な「IP allowlist」に留まらない。ポリシーはfilesystem、process、network destination、binary、provider credential、API method/path等を扱う。RESTの場合はHTTP method/

path、GraphQLではoperation type/name/root field、WebSocketではupgradeおよびtext message等の検査機能が公開されている。 22

ただしここには重要な落とし穴がある。**L7 inspectionを有効化しても `enforcement` のデフォルト値は `audit`** であり、規則違反をログには残すが通信を転送する。実際に遮断するには `enforcement: enforce` が必要である。さらに `protocol` を指定しないendpointはhost/port/binaryまでは検査しても、HTTP method/pathを制限せず、非HTTPトラフィックは内容非検査でrelayする。`tls: skip` ではTLS終端、credential rewrite、L7 inspectionが無効になる。したがって「OpenShellを入れただけで全外部I/Oが意味論レベルで拒否される」と理解するのは誤りである。 22

認証情報管理では、providerがservice名とcredentialを結びつけ、Supervisorがpolicyで許された宛先にだけcredentialを使う。これによりagent process内に長期API keyを置かない設計となる。外向きTLSは通常OpenShellがsandboxごとの一時CAを使って終端するため、credential injectionやL7検査が可能になる。これは強い制御点になる一方、企業利用ではTLS interceptionとしてのプライバシー、証明書管理、アプリ互換性を評価すべきである。 23

**形式手法。** Policy Proverは、候補policyがoperator-defined boundaryより広い権限を許すかを判定する。`within_boundary` のみ成功扱いで、`exceeds_boundary`、`unsupported`、`inconclusive` 等は失敗として扱うよう公式文書が推奨する。RESTについても `enforcement: enforce` のポリシーだけが証明対象になり、audit modeは実際にblockしないためunsupportedとなる。 3

しかし「proof」の意味には限界がある。公式文書自身が、合格とは「Proverがモデル化している部分でboundaryを越えない」という意味に過ぎず、「特定タスクに安全」「実行中sandboxが正しく強制している」「最小権限である」ことまでは意味しないと明記する。また、1,024超のnetwork rules、4,096超のendpoints等ではresource limitにより `inconclusive` になり得る。GraphQL、WebSocket、MCP、JSON-RPCは現行boundary proofの対象外である。これは本プラットフォーム評価で最重要の制約の一つである。 3

**Sentry／BlueField-4層。** NVIDIAによれば、SentryはDOCAを用いてrequest/response、agent interaction、policy decision、tool/data accessを相関し、attested telemetryとagent identity／delegated authorityの確認を行う。Vera Rubin PODではBlueField-4を「nodeからmodelへの唯一の経路」に置き、その位置からhostとは独立してline-speedで観測・policy enforcementを行う設計である。逸脱したagentをミリ秒単位でquarantine可能とされる。 18

ただし、Sentryの公開資料では、**逸脱検出モデルの具体的アルゴリズム、behavioral profileの表現形式、閾値学習方式、誤検知率、攻撃耐性、attestationプロトコルの完全仕様、鍵管理方式、個々のDOCA API、隔離状態機械の詳細は未記載**である。したがってOpenShellのソフトウェア仕様ほど第三者が検証できる状態にはまだない。これは発表直後という時期も考慮すべきだが、セキュリティ保証を評価する際には大きな情報ギャップである。 4

**監査・ログ。** OpenShellではOCSF形式のJSONL exportを有効化でき、機械処理、compliance archival、SIEM integrationに利用できる。現行ドキュメントではOCSF v1.8.0が説明され、sandbox supervisorのログとWindows MXC gatewayログについて、ローカルsinkは日次rotateし直近3ファイルを保持する。長期保管が必要な企業では外部SIEM等への即時転送が前提になる。OpenShell自体のローカルログを「改ざん不能長期監査台帳」とみなすべきではない。 24

**供給網対策。** OpenShell repositoryにはLICENSE、SECURITY、third-party notices、lock files、CI関連文書等が公開され、stable releaseはconformance、upgrade、API compatibility、artifact、security checksを通過したpre-releaseから昇格する運用が明記されている。また実行バイナリに対しては最初に観測したSHA-256 digestをpinし、差し替えをdenyする機構がある。 25

一方で、Open Agent Safety Platform全体として、**SLSAレベル、Sigstore/Rekorによる全artifact署名、SBOM提出義務、MCP server／agent skill／モデルweightのprovenance保証、BlueField firmware supply-chainのエンドツーエンド検証仕様**を一つの包括的保証として定義した公開文書は、今回の調査では確認できなかった。OpenShell自身も、外部から取得するmaterialsのsecurity/integrity/suitability確認は利用者責任と明記する。したがってプラットフォーム導入だけでAI agent supply-chainリスクが自動的に解決するわけではない。 [26]

**関連特許。**「Open Agent Safety Platform」「OpenShell」「Sentry」そのものに直接対応する公開特許番号は、2026年10月1日時点の公式発表・公開資料および今回の検索範囲では **確認できず／未記載** である。ただし、技術的な先行基盤として関連性が高い特許群は存在する。Mellanox Technologies名義のRansomware detection in the memory of the data processing unit 系では、DPUのhostとは別のtrusted domainにhardware-accelerated security serviceを置き、host侵害時にもout-of-bandでmemoryを監視し、SIEM/XDRへtelemetryを送り、必要ならhostを隔離する設計が記述されている。また本文にはNVIDIA DOCA AppShieldとMorpheusが具体例として登場する。これはSentryが採る「host-independent DPU security」の思想と技術的に強い連続性を持つが、**当該特許がSentryを直接クレームしているとの公開根拠はない**。 [27]

## 実装状況・入手性・エコシステム

OpenShellは既に公開OSSであり、GitHub上でApache License 2.0として提供されている。2026年10月1日時点の公開ドキュメントでは0.1.x系のstable release cadenceが示され、stable buildはproduction use向けと位置付けられている。Linux x86-64／Arm64、Apple Silicon macOS＋Docker Desktopがsupported、Windows WSL2はexperimentalである。 [28]

Compute/runtime側はDocker、Podman、Kubernetes、MicroVM等を念頭に置いた抽象化があり、KubernetesではGatewayをHelmで展開し、CNIが `NetworkPolicy` を正しく強制することが要件となる。OpenShellはcloud、hybrid、on-premises、air-gapped環境を対象としているため、NVIDIA cloud専用サービスではない。 [29]

SDKは少なくとも **Python、TypeScript、Go、Rust** が公開されている。CLIでsandbox作成、policy管理、provider設定等が可能で、Policy Proverもstandalone commandとしてCIで実行可能である。公開agent skillsもあり、coding agentにOpenShell CLI、policy記述、gateway/inference routingの操作を教える形で提供される。 [30]

OpenShellのanonymous telemetryは運用上のcategory/countに限定するとNVIDIAは説明しており、sandbox name、hostname、file path、prompt、credential、provider/model name、user content等は収集しないとしている。また環境変数等で無効化でき、compile-outも可能とされる。これはair-gapや機密環境で重要な点であるが、実運用ではsource code／packet capture等で自社検証するのが望ましい。 [26]

一方のSentryは状態が異なる。公式資料ではBlueField-4上で動くreference system design／optional security layerとして説明されるものの、**Sentry単体の一般公開GitHub repository、package名、SDKのダウンロード方法、API reference、ライセンス、価格、SLA、BlueField-4以外で同等のout-of-band保証を実現する仕様は未記載**である。したがって「Open Agent Safety Platform全体がApache 2.0 OSS」と表現するのは不正確であり、Apache 2.0と確認できるのはOpenShellである。 [31]

NVIDIAはVera CPU上でOpenShellを使う場合を最適化対象とし、Veraについて「従来CPUインフラより最大80%高速なsandbox performance」と宣伝している。ただし、比較CPU、sandbox workload、測定条件、tail latency、security policy complexity等を含む再現可能なbenchmark methodologyは対象ページには示されていないため、これは **ベンダー公称値** と評価すべきである。 [32]

導入・統合事例では、SpaceXAIがCursor coding agentsとGrok modelsで利用、AnthropicがClaude Managed AgentsとのOpenShell／BlueField integrationを進め、Scale AIがGenAI Portfolioへ組み込み、Salesforce/NVIDIAがSlackとOpenShellを統合してagent activity、audit event、追加権限approval/denyをSlackから扱えるようにしたと発表されている。さらにSAP、Red Hat、Canonical、SUSE、金融、エネルギー、ロボティクス企業等が協力・統合を表明している。NVIDIA Japanの公式抄訳では「100を超える組織」が活用しているとされる。 [33]

ただし発表直後の段階では、各社について「pilot」「integration」「production」「共同開発」の境界が一様ではないため、ベンダーロゴの数をそのまま商用本番導入数と解釈すべきではない。この点は独立報道でもNVIDIAの広いパートナー網が紹介されている一方、技術効果についてはまだベンダー発表への依存度が高い。 [34]

## 競合・類似技術との比較

比較対象として、OpenAIのAgents API／Agents SDK sandbox、Anthropic Claude Managed Agents／Claude sandboxing、Microsoft Agent Governance Toolkitを採用した。これらは完全に同一カテゴリではなく、Anthropicは実際にNVIDIAと統合しているため「競合かつ補完関係」でもある。 [35]

| 観点 | NVIDIA Open Agent Safety Platform | OpenAI Agents API / Agents SDK | Anthropic Managed Agents / sandbox | Microsoft Agent Governance Toolkit |
|---|---|---|---|---|
| 主眼 | agent実行をモデル外・OS・hardwareまで含めて拘束 | managed harness＋controlled sandboxで長時間agentを実行 | agent loopとsandboxを分離し権限管理 | framework横断runtime governance |
| モデル／ハーネス非依存性 | 高い。open/closed model、任意agent/harnessを想定。 [36] | SDKは柔軟だが、OpenAI managed Agents APIはCodex harness中心。 [37] | Claude中心。Managed Agentsは Anthropic managed/self-hosted sandboxを選択可能。 [38] | framework-agnosticを明示。LangChain、CrewAI、Google ADK、Microsoft Agent Frameworkなどのhookを利用。 [39] |
| Sandbox | OpenShell per-agent sandbox、kernel enforcement | hostedまたはself-hosted sandbox。Agents SDKにもnative sandbox execution。 [40] | sandbox runtimeを利用。Anthropicはruntimeを OSS化したと説明。 [41] | execution sandboxingをtoolkitの一部として提供。 [42] |
| agent process外のpolicy enforcement | Supervisor／Gateway側で実施 | harness・sandbox境界あり。NVIDIA型DPU強制は調査した公開資料では未記載。 [43] | permission policyでtool/MCP callをrun/deny/pause approval。 [44] | deterministic runtime policy enforcement。 [39] |

| 観点 | NVIDIA Open Agent Safety Platform | OpenAI Agents API / Agents SDK | Anthropic Managed Agents / sandbox | Microsoft Agent Governance Toolkit |
|---|---|---|---|---|
| 形式検証 | Policy Proverあり。ただしモデル化範囲に制限。 [3] | 同等のpolicy-boundary形式証明は調査した公開資料で未記載 | 同等機構は調査した公開資料で未記載 | Rego系policy／deterministic enforcementはあるが、NVIDIA型boundary theorem provingは調査資料で未記載。 [42] |
| Credential隔離 | agentからcredentialを隠し、Supervisorが許可宛先に注入。 [15] | 環境／tool構成に依存。公開資料では同一方式を一般保証とはしていない。 [45] | permission／sandboxで制御。詳細は構成依存。 [46] | zero-trust identityをtoolkit要素に含む。 [42] |
| Hardware OOB enforcement | **あり。** BlueField-4/ Sentryが主要差別化点。 [11] | 調査した公開資料では未記載 | 調査した公開資料では未記載 | 調査したAGT公開資料ではBlueField型の専用DPU watchdogは未記載 |
| Host compromise時の独立強制 | Sentryはそこを狙う。 [32] | 公式sandboxで多層隔離を行うが、比較対象資料で同等DPU保証は未記載。 [47] | sandbox boundaryはあるが、同等hardware OOBは未記載 | runtime/OS-level controls中心 |
| Human approval | dangerous policy expansionを人間reviewへ送れる | approvalをSDK/app側で制御。 [45] | `auto` policyがrun/deny/pause for approvalを判断。 [44] | policy workflowとして実装可能 |
| Audit | allow/deny＋OCSF JSON/ SIEM | tracing／platform observabilityを提供するが体系は異なる | tool evaluation event等 | auditをtoolkit構成要素として明示。 [42] |
| OSS | OpenShell: Apache 2.0。Sentry全体は未記載。 [26] | Agents SDK/ Codex harnessにOSS要素。managed infrastructureはサービス。 [48] | sandbox runtime OSS。Managed Agentsはservice。 [41] | MIT License。 [39] |
| Deployment | cloud/hybrid/ on-prem/edge/ air-gap | OpenAI hosted/ self-hosted sandbox等。 [45] | Anthropic-managed/self-hosted sandbox。 [38] | Azure/Kubernetes等を含みframework横断 |

| 観点 | NVIDIA Open Agent Safety Platform | OpenAI Agents API / Agents SDK | Anthropic Managed Agents / sandbox | Microsoft Agent Governance Toolkit |
|---|---|---|---|---|
| 主な公開上の制約 | Prover coverage、L7 audit default、Sentry詳細不足、BF4依存 | OpenAI ecosystem依存が比較的強い | Claude ecosystem中心 | hardware-independentで広いが、NVIDIA型in-silicon watchdogは公開資料にない |

**OpenAIとの差。** OpenAIは2026年のAgents SDKでnative sandbox executionと「harnessとcomputeの分離」を強化し、Agents APIではOpenAI-hosted sandbox、自前sandbox、sandboxなしを選べる。これはNVIDIAの「agent loopとexecution environmentを分離する」という方向と共通する。OpenAIはさらに、自らの研究インシデントを受けて、VM-based isolated sandbox、strict network isolation、多層監視等を強化している。 [49]

差は、NVIDIAがモデル／ハーネスの提供者ではなく **「他社agentを囲う基盤」** として立ち位置を置き、さらにBlueField-4という別シリコンに最終監視点を移している点にある。またPolicy Proverを公開し、policy変更そのものを形式的に境界比較する点も明確な差別化要素である。 [50]

**Anthropicとの差。** AnthropicもClaude Code sandboxingで強い境界を作り、社内利用でpermission promptが84%減ったと報告し、runtimeをopen-source化している。またManaged Agentsはagent loopをsandboxとは別のserverで動かし、server-side permission policyがtool/MCP callを許可・拒否・人間承認待ちにできる。 [51]

ただしこれはNVIDIAと相反するより補完的であり、実際にNVIDIAとAnthropicはManaged AgentsにOpenShell/BlueFieldを重ねる統合を公表している。すなわちAnthropicの「brain/harness側のpermission control」に、OpenShellのOS/runtime boundaryとSentryのhardware boundaryを追加する構図である。 [52]

**Microsoftとの差。** Microsoft Agent Governance ToolkitはMIT LicenseのOSSで、policy、identity、sandboxing、audit等をframework-agnosticに挿入する。Microsoftは全OWASP agentic AI risksを対象とし、sub-millisecondのdeterministic policy enforcementを主張している。Python 3.10+、TypeScript/.NET等にも広がっており、ソフトウェア・ガバナンス層だけを比べるとOpenShellの有力な競合である。 [39]

Microsoft側の強みは既存agent frameworkへの軽量な組み込み性とportable control specificationであり、NVIDIA側の差はOS kernel enforcement、policy proof、credential brokering、そして専用DPUによる独立監視を一つのリファレンス設計に統合した点である。なおMicrosoft自身も、Toolkitが特定法規へのcomplianceを自動保証するものではないと明示している。 [53]

## 運用リスクと導入ベストプラクティス

最初の実務リスクは **「設定すれば安全」ではなく「設定の意味を正確に理解しないとsilent permissiveになる」** ことである。代表例がL7 `audit` defaultである。method/path規則を書いても `enforcement: enforce` にしなければ違反は通過する。またprotocol未指定endpointはL4以上の意味論的制御が弱く、`tls: skip` ではL7 inspectionとcredential rewriteを失う。従ってpolicy reviewでは「allowlistが存在するか」ではなく、「どのレイヤでblockしているか」を確認する必要がある。 [22]

次に **誤検知と回避のトレードオフ** がある。厳密なdeny-by-defaultはデータ持ち出しを抑えるが、正当なagentの新規dependency取得、未知API endpoint、redirect、CDN、dynamic endpoint等を阻害し得る。一方、運用負荷を減らすためwildcard、広いbinary glob、L4-only、`tls: skip` 等を許すと防御強度が下がる。特に公式文書は `/**` のような広いbinary globがbinary-scoped enforcementの目的を損なうと警告している。 20

形式検証にもfalse sense of securityの危険がある。Proverで `within_boundary` でも、「タスクとして安全」「runtime implementationがbug-free」「許可済みAPIが意味論的に安全」ではない。逆に `unsupported` や `inconclusive` を無理に自動承認すればproof layerを事実上無効化する。CI/CDでは **`within_boundary` 以外をfail closed** とし、coverage fieldも確認することが重要である。 3

**性能影響**について、ネットワークの全TCP/DNS要求をSupervisorへ仲介し、必要に応じてTLS terminate、HTTP/GraphQL/WebSocket parse、credential rewrite、audit loggingを行う以上、理論上の追加latency／CPU負荷はゼロではない。NVIDIAはVeraでminimal overhead、Sentryでline-speed、Vera sandboxで最大80%高速等を主張するが、現時点では独立した再現可能ベンチマークを確認できない。導入時には自社ワークロードでp50/p95/p99 latency、throughput、CPU/memory、connection setup、large transfer等を測るべきである。 4

**可用性**では、Supervisor connectionが失われるとsandboxがfreezeするfail-closed設計がセキュリティ上は望ましい反面、Supervisor/Gateway/network control plane障害がagent fleet停止につながる。従って高可用性設計、credential renewal、certificate rotation、failover試験をセキュリティ試験と同じ重要度で扱う必要がある。 54

**ログ**はローカル保存だけでは不十分である。OCSF JSON sinkは直近3ファイル保持という仕様があるため、規制・インシデント調査に必要な長期保持を求める企業はSIEM／object storageへ速やかにexportし、別trust domainに保存する必要がある。Sentryの「attested telemetry」がどの程度tamper-evidentな監査証跡を提供するかの完全仕様は **未記載** である。 55

**法的・倫理的論点**として、OpenShellはTLSをsandbox内CAで終端してrequestを解析できるため、個人情報、顧客データ、通信秘密に相当し得るデータをセキュリティレイヤが処理する可能性がある。またactivity/audit telemetryもagentの業務内容を推測可能な情報になり得る。したがって利用目的、保持期間、閲覧権限、地域間転送、従業員監視との関係などを法務・privacy担当と定義すべきである。これは技術仕様から導かれる運用上の推論であり、NVIDIAが特定法域への適合を自動保証しているという意味ではない。 56

OpenShellのlicenseはApache 2.0だが、GitHub上のdisclaimerは、外部から取得するmaterialsの条件・license・security・integrityを利用者自身が確認すべきであり、softwareは“AS IS”で提供されるとしている。Sentryについての個別商用license、保証、indemnity、SLA、責任分界は今回確認した公開資料では **未記載** である。 26

**推奨導入手順**は、次のような段階的方式が適切である。

**資産・権限モデルの定義。** まずagentごとに必要なfilesystem、executables、network destinations、API methods、data classes、credentials、model endpointsを列挙する。「agentに何をさせたいか」ではなく「最悪、agentが侵害された場合に何を許してよいか」を基準にboundaryを決める。AgentDojo等が示すように、モデルがprompt injectionを受ける前提で権限設計するのが安全側である。 57

**OpenShell単体でのPoC。** 非本番でdeny-by-defaultから開始し、Docker／PodmanまたはKubernetesでagentをsandbox化する。rootは禁止し、Landlock/seccomp要件を満たすkernelを使用する。Windows WSL2はexperimentalであるため、高保証productionの基準環境には現時点ではLinuxを優先すべきである。 28

**ネットワークポリシーをL7まで狭める。** host/port allowlistだけでなく、可能なAPIは `protocol: rest` 等を設定し、method/pathを絞る。初期learning periodでは `audit` で通信パターンを収集してもよいが、本番移行前に必要箇所を `enforce` に切り替える。credentialed endpointで `tls: skip` や `allow_uninspected_credentials` を使う場合は例外申請制にする。 19

**Credentialをagentから排除。** API key等をimage/environment variableに焼き込まず、provider＋Supervisor経由でinjectする。Credentialごとにdestination bindingを作り、同じAPI keyを複数agentや広いendpointで共有しない。 15

**Policy ProverをCI gate化。** 変更前policyをboundaryと比較し、`within_boundary` 以外はdeploy不可にする。特に `coverage.domains` を検査し、GraphQL/MCP等のunsupported領域を「proof済み」と誤認しない。unsupported部分は別のstatic test／integration test／human reviewで補完する。 3

**敵対的テスト。** Prompt injection、tool poisoning、credential extraction、DNS rebinding、redirect、cloud metadata access、shell escape、binary replacement、shared-service pivot、Supervisor disconnect等を試す。特にOpenAIの2026年インシデントが示したように「direct Internet accessなし」だけでは不十分なので、共有registry、artifact server、package mirror、CI、private networkを通るtransitive pathも攻撃対象としてテストすべきである。 58

**監査基盤を先に接続。** OCSF JSONを有効化し、SIEMに転送してpolicy allow/deny、credentialed access、unexpected DNS、policy expansion、sandbox restart等にalertを設ける。ローカル3世代だけに依存しない。 24

**高リスク環境でSentryを追加。** 金融、重要インフラ、frontier-model eval、長時間自律agent、物理ロボットなど、host compromiseを含むthreat modelを持つ場合はBlueField-4/Sentryを追加検討する。ただし導入前にNVIDIAからSentryの具体的support matrix、DOCA version、firmware dependency、attestation/key management、failure mode、performance data、licensingを取得することが必要である。公開資料だけではこれらの多くが未記載である。 18

**インシデント対応runbookを事前作成。** quarantineを単にkillとせず、agent identity、sandbox ID、policy version、credential use、network flow、tool calls、model interaction、artifactsを保存してからcredential revoke／sandbox destroy／image rebuildへ移行できるようにする。Sentryの隔離操作と企業SIEM/SOARをどう連携させるかは、公開仕様で詳細が不足しているためPoCで確認が必要である。 55

## 今後の展望・研究課題

第一の研究課題は、**形式検証の対象拡大**である。現在のProverはL4 TCP/REST等に強い一方、GraphQL、WebSocket、MCP、JSON-RPCはboundary proofではunsupportedである。今後のagent ecosystemではMCPやevent-stream型protocolの比重が増すため、「HTTP接続は許可された」というL4/L7証明から、「このtool semantic operationだけが許可された」というcapability-level proofへ発展させる必要がある。 59

第二は、**policy proofとruntime enforcementのend-to-end整合性証明**である。現在の公式仕様も、Prover成功は「running sandbox actually enforces it」ことまで証明しないと明記する。将来的にはpolicy compiler、Supervisor、seccomp/Landlock setup、NetworkPolicy、DPU enforcementまで同じmachine-checkable specificationから生成し、remote attestationで実際の配置を証明する方向が望ましい。 17

第三は、**behavioral drift detectionの科学的評価**である。Sentryはbehavioral profileを用いて逸脱を検出するとされるが、何をfeatureにし、どの時間窓で、どのfalse-positive／false-negative tradeoffを持つかは未公開である。AgentDojoやAutoDojoのような動的・生成型benchmarkに、「runtime policy bypass」

「shared-service pivot」「credential misuse」「long-horizon drift」を加え、Sentry/OpenShellを含むシステム単位の評価セットを構築する価値が高い。 60

第四は、**agent identityとdelegated authorityの標準化**である。Sentry/DOCAはagent identityと委任権限の連続検証を掲げるが、agent → subagent → MCP server → SaaS APIと権限が委任される際のportable identity、capability token、revocation、lineage規格は業界横断で未成熟である。NVIDIA自身が「labs、enterprises、hardware providersのshared responsibility」を原則として掲げており、この層はOpen Secure AI Alliance等で標準化する余地がある。 18

第五は、**供給網セキュリティの統合**である。将来的にはsandbox imageだけでなく、agent skill、MCP server、tool schema、model adapter、policy bundle、DOCA component、firmwareまでSBOM、署名、provenance、reproducible buildを連結し、実行時にattestationで検査することが望ましい。OpenShellのbinary SHA-256 pinningは有用だが、これは全供給網のprovenance保証とは異なる。 61

第六は、**性能・可用性の標準benchmark**である。「up to 80% faster」「line speed」「milliseconds」といったベンダー表現を、policy rule数、connection rate、TLS inspection、REST parsing、large payload、multi-agent concurrency、failover等を統一したbenchmarkで検証する必要がある。安全性のための追加制御が高負荷時にfail-openへ変化しないことも評価項目にすべきである。 4

第七は、**物理AIへの拡張**である。NVIDIAはroboticsまでOpen Agent Safety Platformの対象に含めるが、ネットワークAPI拒否と物理 actuator safetyは同一ではない。ロボットではvelocity、force、workspace、human proximity等の時間制約を満たすhard real-time interlockが必要になる。現行公開資料では、OpenShell/Sentryとfunctional safety規格・robotics safety controllerをどのように統合するかの詳細は **未記載** であり、今後の重要テーマである。 62

## 重要技術用語

| 用語 | 定義 |
|---|---|
| **AI Agent** | モデル推論だけでなく、ファイル操作、shell、API、database、tool等を自律的に呼び出し、複数stepの目的を遂行するソフトウェア主体。 |
| **Agent Harness** | モデル呼び出し、context、tool execution、loop、subagent等を調整するagent実行ロジック。OpenShellは原則としてその外側に境界を置く。 36 |
| **OpenShell** | NVIDIAのApache 2.0 OSS secure runtime。sandbox、kernel enforcement、Supervisor、Gateway、policy、provider、Policy Prover等からなる。 63 |
| **Sentry** | BlueField-4上のout-of-band watchdogとして設計されたNVIDIAのagent監視・強制層。 52 |
| **DPU** | Data Processing Unit。ネットワーク／インフラ／セキュリティ処理をホストCPUから分離して実行できるプロセッサ。SentryではBlueField-4が用いられる。 11 |
| **Out-of-Band Enforcement** | agent／hostの通常実行環境から独立した信頼ドメインで監視・強制し、host compromiseで同時に無効化されにくくする設計。 32 |
| **Supervisor** | sandbox外の信頼側プロセス。policy check、DNS、credential injection、実通信の代理実行を担う。 15 |
| **Gateway** | OpenShell control plane。認証、sandbox lifecycle、policy、provider、settings、access等を管理する。 15 |

| 用語 | 定義 |
|---|---|
| **Landlock** | Linux kernelのunprivileged filesystem sandboxing機構。OpenShellがfilesystem制限に利用する。 16 |
| **seccomp user notification** | syscallをkernel側で捕捉し、userspaceの信頼プロセスに判断を委ねるLinux機構。OpenShellではnetwork mediationに用いられる。 16 |
| **Policy Prover** | policy変更がoperator-defined boundaryを超えるか形式的に検査するOpenShell component。 17 |
| **Default Deny** | 明示的に許可されていない操作を拒否する原則。OpenShellのネットワーク／アクセス設計の基礎。 36 |
| **L4 / L7 Inspection** | L4はhost/port等のtransport layer制御、L7はHTTP method/pathやGraphQL operation等のapplication protocol内容まで見る制御。 20 |
| **Credential Brokering** | agentへsecretを直接渡さず、信頼側proxyが許可された外部要求にだけ資格情報を付加する方式。 15 |
| **OCSF** | Open Cybersecurity Schema Framework。OpenShellは監査イベントをOCSF JSONとしてexport可能。 24 |
| **Drift** | agent actionが意図されたtaskまたはoperating constraintsから逸脱すること。NVIDIAはbug、policy block、曖昧な指示、長時間実行等を原因例に挙げる。 11 |
| **Fail Closed** | control component障害時に安全制御を解除するのではなくworkloadを停止／freezeする設計。OpenShellはSupervisor channel切断時にこの方式を取る。 16 |
| **Attested Telemetry** | 信頼性・真正性を検証可能にすることを意図したtelemetry。Sentry/DOCAが提供するとNVIDIAは説明するが、詳細protocolは公開資料では未記載。 52 |

## 主要情報源・URL・発見日

以下は、本調査で特に重視した一次・準一次資料である。**発見日はいずれも2026年10月1日**。日本語公式資料を先頭に置いた。

**NVIDIA Japan公式抄訳 — Open Agent Safety Platform発表**
発見日：2026-10-01
URL: `https://blogs.nvidia.co.jp/blog/open-agent-safety-platform/`
短い抜粋：「OpenShell やスキルを含む…ソフトウェアは…GitHub から入手可能」
米国発表の日本語公式抄訳で、国内向けの提供状況・パートナー情報を確認する上で最優先の日本語資料。 64

**NVIDIA Newsroom — NVIDIA Launches Open Agent Safety Platform**
発見日：2026-10-01
URL: `https://nvidianews.nvidia.com/news/open-agent-safety-platform`
短い抜粋：“OpenShell open source software and the NVIDIA Sentry reference system design”
2026年9月28日の一次発表。構成要素、Sentry、採用企業、提供状況の基準資料。 10

**NVIDIA Technical Blog — Continuous In-Silicon Agent Monitoring**
発見日：2026-10-01

URL: `https://developer.nvidia.com/blog/nvidia-open-agent-safety-platform-a-reference-for-continuous-in-silicon-agent-monitoring/`
短い抜粋：“Enforcement must be out of band”
三層アーキテクチャ、五つの設計原則、drift、Vera/BlueField配置を説明する中核技術資料。 65

**NVIDIA Open Agent Safety Platform製品ページ**
発見日：2026-10-01
URL: `https://www.nvidia.com/en-us/solutions/ai/agent-safety/`
短い抜粋：“host-independent monitoring and enforcement”
OpenShell、Sentry、BlueField、Veraの役割分担と性能公称値を掲載。 32

**NVIDIA OpenShell製品ページ**
発見日：2026-10-01
URL: `https://www.nvidia.com/en-us/ai/openshell/`
短い抜粋：“Deny access by default”
sandbox、Gateway、Supervisor、Policy Proverの公式概要。 36

**OpenShell公式GitHub repository**
発見日：2026-10-01
URL: `https://github.com/NVIDIA/OpenShell`
短い抜粋：“licensed under the Apache License 2.0”
OSS実装、SDK、CLI、telemetry、license、security policyを確認できる一次ソース。 66

**OpenShell Architecture Documentation**
発見日：2026-10-01
URL: `https://docs.nvidia.com/openshell/latest/about/architecture`
短い抜粋：“The supervisor is trusted and makes the decisions.”
通信経路、fail-closed、Landlock/seccomp、Supervisor/Gateway trust boundaryの最重要仕様。 2

**OpenShell Policy Prover Documentation**
発見日：2026-10-01
URL: `https://docs.nvidia.com/openshell/latest/how-it-works/policies/prover`
短い抜粋：“Only `within_boundary` means that the check passed.”
形式検証の機能と、unsupported/inconclusiveを含む限界を明示する一次文書。 3

**OpenShell Security Best Practices**
発見日：2026-10-01
URL: `https://docs.nvidia.com/openshell/latest/security/best-practices`
短い抜粋：“Default `audit`.”
L4/L7、TLS、SSRF、binary hash binding等の運用上の重要な注意点を確認した。 22

**OpenShell OCSF JSON Export**
発見日：2026-10-01
URL: `https://docs.nvidia.com/openshell/latest/observability/ocsf-json-export`
短い抜粋：“retains the three most recent files”
監査ログ、SIEM integration、保存期間の仕様を確認。 24

**OpenShell Support Matrix**
発見日：2026-10-01
URL: `https://docs.nvidia.com/openshell/latest/about/support-matrix`

短い抜粋：“Use a stable release for production deployments.”
OS、architecture、release qualification、production support条件の一次資料。 [67]

**関連特許 — Ransomware detection in the memory of the data processing unit**
発見日：2026-10-01
URL: `https://patents.google.com/patent/CN116595521A/en`
短い抜粋：“out-of-band security software in a separate and isolated trusted domain”
Mellanox Technologies名義。DPU上の独立trusted domain、host monitoring、隔離、NVIDIA DOCA AppShield等を記載。Open Agent Safety Platformとの直接の特許対応関係は **未記載**。 [68]

**AgentDojo — Debenedetti et al.**
発見日：2026-10-01
URL: `https://arxiv.org/abs/2406.13352`
短い抜粋：“vulnerable to prompt injection attacks”
ツール利用agentに対するprompt injectionを体系的に評価する代表的学術研究。 [69]

**CaMeL: Defeating Prompt Injections by Design**
発見日：2026-10-01
URL: `https://arxiv.org/abs/2503.18813`
短い抜粋：“creates a protective system layer around the LLM”
モデル外のcontrol/data-flow分離とcapabilityによる防御を提案し、OpenShellとの設計上の比較に有用。 [70]

**OpenAI — Agents SDK の進化（日本語）**
発見日：2026-10-01
URL: `https://openai.com/ja-JP/index/the-next-evolution-of-the-agents-sdk/`
短い抜粋：「ネイティブのサンドボックス実行」
OpenAIのharness／sandbox分離との競合比較に使用。 [71]

**OpenAI — Agents API**
発見日：2026-10-01
URL: `https://developers.openai.com/api/docs/guides/agents`
短い抜粋：“OpenAI hosted sandbox, self-hosted sandbox, or no sandbox”
managed/self-hosted execution modelの比較に使用。 [45]

**OpenAI–Hugging Face Incident Technical Report**
発見日：2026-10-01
URL: `https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf`
短い抜粋：“strong sandboxing and isolation mechanisms”
共有サービス等を介するtransitive escape pathと多層隔離の必要性を示した一次インシデント報告。 [47]

**Anthropic — How we contain Claude across products**
発見日：2026-10-01
URL: `https://www.anthropic.com/engineering/how-we-contain-claude`
短い抜粋：“open-sourced the runtime”
Anthropicのsandbox boundaryとの比較に使用。 [41]

**Anthropic — Claude Platform Release Notes**
発見日：2026-10-01

URL: `https://docs.anthropic.com/en/release-notes/api`
短い抜粋：“runs it, denies it, or pauses for your approval”
Managed Agentsのtool/MCP permission enforcementを確認。 44

**Microsoft — Agent Governance Toolkit**
発見日：2026-10-01
URL: `https://opensource.microsoft.com/blog/2026/04/02/introducing-the-agent-governance-toolkit-open-source-runtime-security-for-ai-agents/`
短い抜粋：“open-source project … MIT license”
runtime governance、framework-agnostic integration、OSS licenseの比較基準。 39

**独立報道 — Reuters Breakingviews**
発見日：2026-10-01
URL: `https://www.reuters.com/commentary/breakingviews/jensen-huangs-invisible-hand-is-tied-ai-safety-2026-09-28/`
NVIDIAの発表を外部視点から位置付け、SentryのBlueField依存等を確認する補助資料として利用。 72

**独立報道 — Axios**
発見日：2026-10-01
URL: `https://www.axios.com/2026/09/28/nvidia-ai-agent-safety`
発表直後の業界文脈と「milliseconds」の外部報道確認に使用。 73

総じて、2026年10月1日時点で最も検証可能性が高いのは **OpenShell** であり、コード、policy semantics、kernel boundary、Prover、loggingまで比較的詳細に公開されている。一方で **Sentryは戦略的に最も差別化された部分であると同時に、技術詳細が最も不足している部分**でもある。したがって企業評価では、「OpenShellで既に検証可能な保証」「SentryについてNVIDIAが公称している保証」「まだ公開検証できない保証」を明確に分離することが、本プラットフォームを過大評価も過小評価もしないための最重要原則となる。 74

---


1 4 7 10 18 33 52 62 NVIDIA Launches Open Agent Safety Platform to Secure Agents From Testing to Deployment | NVIDIA Newsroom
https://nvidianews.nvidia.com/news/open-agent-safety-platform

2 13 15 16 23 50 54 Architecture | NVIDIA OpenShell
https://docs.nvidia.com/openshell/latest/about/architecture

3 9 17 59 Policy Prover | NVIDIA OpenShell
https://docs.nvidia.com/openshell/latest/how-it-works/policies/prover

5 11 31 60 65 NVIDIA Open Agent Safety Platform: A Reference for Continuous In-Silicon Agent Monitoring | NVIDIA Technical Blog
https://developer.nvidia.com/blog/nvidia-open-agent-safety-platform-a-reference-for-continuous-in-silicon-agent-monitoring/

6 47 58 https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf
https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf

8 35 37 43 45 Agents API - OpenAI Developers
https://developers.openai.com/api/docs/guides/agents

12 29 36 NVIDIA OpenShell | Open, Secure Runtime for AI Agents
https://www.nvidia.com/en-us/ai/openshell/

14 19 20 22 61 OpenShell Security Best Practices for Controls, Risks, and Configuration Guidance | NVIDIA OpenShell
https://docs.nvidia.com/openshell/latest/security/best-practices

21 57 69 AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents
https://arxiv.org/abs/2406.13352

24 55 56 OCSF JSON Export | NVIDIA OpenShell
https://docs.nvidia.com/openshell/latest/observability/ocsf-json-export

25 28 67 Support Matrix | NVIDIA OpenShell
https://docs.nvidia.com/openshell/latest/about/support-matrix

26 30 63 66 74 GitHub - NVIDIA/OpenShell: OpenShell is the safe, private runtime for autonomous AI agents. · GitHub
https://github.com/NVIDIA/openshell

27 68 CN116595521A - Ransomware detection in the memory of the data processing unit - Google Patents
https://patents.google.com/patent/CN116595521A/en

32 NVIDIA Open Agent Safety Platform: Secure AI Agents
https://www.nvidia.com/en-us/solutions/ai/agent-safety/

34 73 Nvidia says new tool can contain rogue AI agents in "milliseconds"
https://www.axios.com/2026/09/28/nvidia-ai-agent-safety

38 Agent SDK overview - Claude Code Docs
https://docs.anthropic.com/en/docs/claude-code/sdk

39 Introducing the Agent Governance Toolkit: Open-source runtime security for AI agents | Microsoft Open Source Blog
https://opensource.microsoft.com/blog/2026/04/02/introducing-the-agent-governance-toolkit-open-source-runtime-security-for-ai-agents/

40 49 The next evolution of the Agents SDK | OpenAI
https://openai.com/index/the-next-evolution-of-the-agents-sdk/

41 51 How we contain Claude across products \ Anthropic
https://www.anthropic.com/engineering/how-we-contain-claude

42 Agent Control Specification: Portable runtime governance for AI ...
https://commandline.microsoft.com/agent-control-specification-runtime-governance/

44 Claude Platform release notes
https://docs.anthropic.com/en/release-notes/api

46 Configure permissions - Claude Code Docs
https://docs.anthropic.com/en/docs/claude-code/permissions

48 New tools for building agents | OpenAI
https://openai.com/index/new-tools-for-building-agents/

53 Build agents you can trust across any framework with open ...
https://devblogs.microsoft.com/foundry/build-2026-open-trust-stack-ai-agents/

64 NVIDIA、テストから展開までエージェントを保護する Open ...
https://blogs.nvidia.co.jp/blog/open-agent-safety-platform/

70 Defeating Prompt Injections by Design
https://arxiv.org/abs/2503.18813

71 Agents SDK の進化 - OpenAI
https://openai.com/ja-JP/index/the-next-evolution-of-the-agents-sdk/

72 Jensen Huang's invisible hand is tied on AI safety
https://www.reuters.com/commentary/breakingviews/jensen-huangs-invisible-hand-is-tied-ai-safety-2026-09-28/