Hades Blackwatch Detailed Full-Stack Analysis

HB HADES BLACKWATCH

Internal - Version 2.2

Executive Summary

Target: Hades Blackwatch Skill Repository

Date: 2026-05-28

Request changes: high-risk issues require remediation before approval.

Critical	High	Medium	Low	Informational
0	1	2	3	0

Risk posture: Elevated. This report contains 6 normalized finding(s), ordered by severity and supported by local source review and generated evidence.

Scope and Authorization

ENVIRONMENT
local repository and generated documentation/report artifacts

ACCESS LEVEL
source code, docs, examples, local script execution, generated report artifacts

AUTHORIZED
True

INCLUDED
[
  "SKILL.md full-stack workflow",
  "agent manifest and specialist role files",
  "MCP escalation-only policy",
  "scanner normalization pipeline",
  "MCP manifest audit pipeline",
  "surface mapping pipeline",
  "prompt attack pack generation",
  "HTML/PDF report generation",
  "README and docs/index.html",
  "example scanner and report inputs"
]

EXCLUDED
[
  "live MCP server calls",
  "network-based scanner integrations",
  "third-party package vulnerability lookups",
  "active tests against external applications",
  "production deployment or installation review"
]

Methodology

FRAMEWORKS
[
  "OWASP LLM Top 10",
  "OWASP Agentic Top 10",
  "OWASP Agentic Skills Top 10",
  "OWASP AIVSS",
  "MITRE ATLAS",
  "NIST AI RMF GenAI Profile",
  "OWASP Web/API Top 10 and CWE for conventional appsec mapping"
]

AGENTS USED

[
"Security Orchestrator",
"Latest Threat Intel",
"AI Threat Model",
"MCP Server Auditor",
"Scanner Registry",
"Prompt Injection",
"Agent Tool Security",
"RAG and Data Security",
"Secure Code Review",
"Supply Chain",
"Finding Correlator",
"Test Builder",
"Report"
]

SCANNER BACKENDS

[
"surface_mapper.py local run",
"mcp_manifest_audit.py local run",
"normalize_findings.py local run",
"prompt_attack_pack.py local run",
"generate_report.py local run",
"manual source review"
]

MCP POLICY

No live MCP calls were used. The MCP necessity test did not pass because local source, docs, examples, generated artifacts, and bundled scripts were sufficient.

Skill Coverage and Specialist Notes

This detailed review is personalized for the Hades Blackwatch skill repository. The specialist notes separate the agents that materially contributed from agents whose domain was checked but not deeply exercised because the repo is a local skill/tooling package rather than a deployed AI application.

Security Orchestrator

Status: Used

Focus: Scope, authorization, MCP necessity, final severity, and report structure

Coverage: Kept the review local and source-driven, avoided unnecessary MCP calls, and prioritized six findings by report-leakage, evidence integrity, MCP audit depth, schema migration compatibility, documentation accuracy, and tests.

Notes: Keep this role as the default entry point for full-stack review so every generated report explains what was actually tested and what was not.

Latest Threat Intel Agent

Status: Used for report refresh

Focus: Current OWASP, MITRE, NIST, and MCP guidance relevant to skills, agent tools, scanner evidence, and MCP boundaries

Coverage: Added dated threat-intelligence signals and official source links so the report explains why redaction, MCP least privilege, evidence provenance, and governance controls matter for this repo.

Notes: Run this agent when reports claim current framework coverage, CVEs, vendor tool behavior, or external threat activity.

AI Threat Model Agent

Status: Used

Focus: Trust boundaries between prompts, local scripts, generated reports, scanner evidence, and future MCP tools

Coverage: Identified the main local attack paths: untrusted scanner evidence flowing into reports, generated reports polluting future analysis, and MCP schema under-classification.

MCP Server Auditor

Status: Used

Focus: MCP escalation policy and manifest risk-tier analysis

Coverage: Validated that no live MCP call was needed and found that nested tool schemas need recursive inspection before real Burp, Qualys, Sonar, or custom MCP adapters are trusted.

Scanner Registry Agent

Status: Partially covered

Focus: Scanner backend claims, evidence ingestion, and auto-detection expectations

Coverage: Confirmed report ingestion and local scripts exist, while scanner auto-detection remains roadmap-only and should be described that way until implemented.

Prompt Injection Agent

Status: Scoped

Focus: Prompt and report-content poisoning paths

Coverage: No live model target was present, but generated artifacts and scanner evidence were treated as untrusted context that can influence later agent runs.

Agent Tool Security Agent

Status: Used

Focus: Tool permissions, evidence-to-report flow, and state-changing MCP policy

Coverage: Mapped the most important tool risks to evidence disclosure, MCP authority classification, and approval boundaries for future scanner adapters.

RAG and Data Security Agent

Status: Low relevance

Focus: Retrieved content and data-store exposure

Coverage: The repository does not implement a RAG runtime, but the same context-poisoning concern appears in generated reports being re-ingested by later local scans.

Secure Code Review Agent

Status: Used

Focus: Python scripts, JSON normalization, HTML/PDF rendering, and local workflow integrity

Coverage: Found redaction, generated-artifact scanning, schema migration, and missing automated test risks.

Supply Chain Agent

Status: Scoped

Focus: Skill, script, schema, and scanner-adapter supply-chain integrity

Coverage: Highlighted schema migration compatibility and future scanner-adapter expectations; live package vulnerability lookup was excluded from this local review.

Finding Correlator

Status: Used

Focus: Deduplication, severity calibration, and root-cause grouping

Coverage: Grouped the findings into three practical themes: sensitive evidence handling, evidence quality/provenance, and implementation-to-documentation alignment.

Test Builder

Status: Used

Focus: Regression tests, fixtures, and release gates

Coverage: Converted each finding into a concrete test or checklist item, with emphasis on redaction, surface exclusion, recursive MCP schema checks, schema identity, and CI coverage.

Threat Intelligence

As of: 2026-05-28

The current threat context for Hades Blackwatch centers on untrusted evidence flowing through agentic review pipelines, MCP/tool authority boundaries, skill and adapter supply-chain integrity, and governance-grade traceability. These signals are mapped to the local findings instead of being generic framework text.

Scanner and tool output is hostile evidence until minimized

Why it matters: OWASP LLM guidance continues to emphasize sensitive information disclosure and improper output handling. For this repo, the practical risk is raw Burp, Qualys, Sonar, Semgrep, CodeQL, Trivy, or custom JSON evidence being copied into polished reports.

Recommended action: Add recursive redaction before normalization and rendering, then make report generation tests assert that secrets never survive into HTML or PDF bytes.

Linked findings: HB-DET-001

MCP security depends on least privilege and precise authorization boundaries

Why it matters: MCP security guidance calls out token passthrough, SSRF, local server compromise, session handling, and scope minimization. For Hades Blackwatch, this makes recursive manifest inspection and escalation-only MCP use important before connecting Burp, Qualys, Sonar, or internal scanners.

Recommended action: Upgrade nested schema analysis, require risk-tier evidence for every MCP tool, and keep live MCP calls behind the necessity test.

Linked findings: HB-DET-003, HB-DET-005

Generated reports can become context poisoning in later agent runs

Why it matters: Agentic systems are sensitive to stale or attacker-shaped context. The local surface mapper currently treats reports as review evidence, which can anchor later analysis on old findings and inflated keyword counts.

Recommended action: Exclude generated report paths by default and require explicit opt-in when report artifacts themselves are the audit target.

Linked findings: HB-DET-002

Skill supply-chain integrity needs stable schemas and regression gates

Why it matters: OWASP Agentic Skills and NIST AI RMF style governance both push toward provenance, traceability, and repeatable controls. Undocumented schema migration and manual-only smoke tests weaken downstream trust in this skill package.

Recommended action: Document schema migration behavior and add automated tests plus CI before new adapters are added.

Linked findings: HB-DET-004, HB-DET-006

Threat Intel Sources

OWASP Top 10 for LLM and GenAI Apps 2025 - https://genai.owasp.org/llm-top-10/ - Prompt injection, sensitive information disclosure, improper output handling, excessive agency, and vector/embedding risks.

OWASP Top 10 for Agentic Applications - https://genai.owasp.org/2025/12/09/owasp-top-10-for-agentic-applications-the-benchmark-for-agentic-security-in-the-age-of-autonomous-ai/ - Agent goal hijack, tool misuse, identity and privilege abuse, agentic supply chain, memory/context poisoning, and inter-agent communication risk.

OWASP Agentic Skills Top 10 - https://owasp.org/www-project-agentic-skills-top-10/ - Skill/plugin execution-layer risks such as malicious skills, over-privilege, insecure metadata, weak isolation, poor scanning, and governance gaps.

Model Context Protocol Security Best Practices - https://modelcontextprotocol.io/docs/tutorials/security/security_best_practices - MCP confused deputy, token passthrough, SSRF, session hijacking, local server compromise, and scope minimization.

NIST AI Risk Management Framework and GenAI Profile - https://www.nist.gov/itl/ai-risk-management-framework - Governance and lifecycle risk management for generative AI systems.

MITRE ATLAS - https://atlas.mitre.org/ - Threat-informed mapping for adversarial tactics and techniques against AI-enabled systems.

Personalized Recommendations

These actions are tailored to Hades Blackwatch Skill Repository, the authorization model, actual evidence sources, and the current finding set.

High

Make evidence redaction the next release gate before this skill is used on real scanner exports.

Why: Hades Blackwatch is designed to ingest Burp, Qualys, Sonar, Semgrep, CodeQL, Trivy, and custom evidence. Without deterministic minimization, the report generator can turn private scanner data into a broadly shared HTML/PDF artifact.

Owner: Security tooling owner

Timeframe: 0-7 days

Linked findings: HB-DET-001

Medium

Keep generated reports out of default source mapping so later reviews stay anchored to primary evidence.

Why: This repository generates rich security reports under reports/. Letting future scans read those files as source evidence can inflate surface counts and reintroduce stale findings.

Owner: Security tooling owner

Timeframe: 1 sprint

Linked findings: HB-DET-002

Medium

Upgrade MCP manifest auditing before connecting live Burp, Qualys, Sonar, or internal scanner MCP servers.

Why: The skill's strategy is MCP only when absolutely needed. That policy is strongest when the auditor can see nested command, URL, credential, body, header, and action fields before a tool is trusted.

Owner: Agent platform / MCP owner

Timeframe: 1 sprint

Linked findings: HB-DET-003

Medium

Document the normalized finding schema migration before downstream integrations depend on it.

Why: The skill now emits the Hades Blackwatch schema after the rename, but downstream consumers may still have older Cerberus-era expectations. A migration note keeps reports, dashboards, and future agents aligned.

Owner: Integration owner

Timeframe: 1 sprint

Linked findings: HB-DET-004

Low

Adjust README and docs language so scanner detection is described as designed support until the registry script exists.

Why: The skill should feel powerful, but security users need exact evidence. Capability wording should distinguish implemented scripts, report ingestion, role guidance, and roadmap work.

Owner: Documentation owner

Timeframe: Next hardening cycle

Linked findings: HB-DET-005

Low

Turn the current smoke checks into committed tests and a CI gate before adding more integrations.

Why: Hades Blackwatch is becoming reusable security infrastructure. Redaction, schema naming, report rendering, generated-output exclusions, and recursive MCP checks should fail fast when changed.

Owner: Engineering quality owner

Timeframe: Next hardening cycle

Linked findings: HB-DET-006

Findings Index

High	**HB-DET-001: Raw scanner evidence can be copied into shareable HTML/PDF reports without secret redaction**
Medium	**HB-DET-003: MCP manifest auditor can under-classify nested permissive tool schemas**
Medium	**HB-DET-002: Surface mapper scans generated reports and previous report inputs, polluting later analysis**
Low	**HB-DET-005: Documentation describes scanner auto-detection before the registry implementation exists**
Low	**HB-DET-004: Hades rename changed the normalized finding schema without compatibility notes**
Low	**HB-DET-006: Smoke checks are manual and not enforced as automated regression tests**

HB-DET-001

High

1. Raw scanner evidence can be copied into shareable HTML/PDF reports without secret redaction

Confidence: high

Affected component: scanner evidence normalization and report generation

Affected file: scripts/normalize_findings.py:103; scripts/generate_report.py:211; scripts/generate_report.py:406

Affected endpoint: Not provided

Affected asset: HTML/PDF security reports generated from scanner exports

Evidence


```
{
  "summary": "The generic normalizer assigns the entire scanner item to the normalized evidence field. The report generator then renders the evidence field into HTML and PDF. HTML escaping is present, but no recursive redaction or evidence minimization is applied before report generation.",
  "observed_code_paths": [
    "normalize_generic stores evidence: item",
    "render_html renders finding evidence through html_block",
    "flatten_for_pdf serializes finding evidence into PDF text"
  ],
  "positive_control": "HTML content is escaped, reducing script injection risk. This does not prevent disclosure of sensitive scanner evidence."
}
```


Attack Path

A Burp, Qualys, Sonar, or custom scanner export includes request headers, cookies, bearer tokens, API keys, session identifiers, request bodies, or response snippets. normalize_findings.py copies the raw item into the evidence field. generate_report.py embeds that evidence in a polished report. The report is then shared more broadly than the raw scanner export, leaking sensitive operational details.

AI / Agent Relevance

The skill explicitly treats scanner output as untrusted input, and the report agent turns that input into downstream deliverables. In an agentic workflow, this expands the blast radius of sensitive scanner evidence unless redaction is deterministic.

Threat Intelligence

{
  "as_of": "2026-05-28",
  "summary": "Current AI security guidance treats tool output, scanner output, and report artifacts as sensitive context that can leak credentials or operational details when copied without minimization.",
  "sources": [
    {
      "name": "OWASP Top 10 for LLM and GenAI Apps 2025",
      "url": "https://genai.owasp.org/llm-top-10/",
      "relevance": "Sensitive information disclosure and improper output handling are directly relevant to report evidence leakage."
    },
    {
      "name": "NIST AI RMF Generative AI Profile",
      "url": "https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence",
      "relevance": "Supports governance-grade traceability, risk measurement, and risk management for GenAI outputs."
    }
  ],
  "signals": [
    {
      "theme": "Evidence exfiltration through polished reports",
      "relevance": "Scanner exports commonly contain request headers, tokens, session values, response snippets, and internal hostnames.",
      "recommended_action": "Add recursive redaction and evidence-minimized report modes before supporting live scanner exports."
    }
  ],
  "mapped_techniques": {
    "mitre_atlas": "Exfiltration and credential access through tool-output handling",
    "cwe": "CWE-200; CWE-532"
  },
  "detection_or_test": "Use fixtures with nested secrets and assert normalized JSON, HTML, and PDF bytes do not contain secret values.",
  "freshness": "Checked against official sources on 2026-05-28",
  "confidence": "High"
}

Framework Mappings

{
  "owasp_llm": "Sensitive Information Disclosure",
  "owasp_agentic": "Information Disclosure through Tool and Context Handling",
  "mitre_atlas": "Credential Access / Exfiltration through tool-output handling",
  "nist_ai_rmf": "Map, Measure, Manage",
  "cwe": "CWE-200; CWE-532",
  "cve": null
}

Score

```
{
  "system": "AIVSS",
  "value": "High",
  "rationale": "The issue can expose high-value security evidence and credentials, is easy to reproduce with realistic scanner exports, and is amplified by automated report sharing."
}
```


Recommended Fix

Add a recursive redaction layer before storing normalized evidence and before rendering reports. Redact key names and values matching Authorization, Cookie, Set-Cookie, token, secret, password, api_key, bearer, private_key, session, credential, client_secret, and configurable organization patterns. Prefer a safe evidence summary allowlist over storing full raw scanner items by default.

Regression Test

Create a fixture with nested request and response objects containing Authorization, Cookie, Set-Cookie, api_key, password, bearer token, private key markers, and session IDs. Run normalize_findings.py and generate_report.py, then assert none of the secret values appear in normalized JSON, HTML, or PDF bytes.

Residual Risk

Some scanner proof data can remain sensitive even after key-based redaction. High-risk reports should carry a classification label and support an evidence-minimized mode.

Attack Path Visualization

For Hades Blackwatch Skill Repository, Raw scanner evidence can be copied into shareable HTML/PDF reports without secret redaction can be understood as a control failure where untrusted influence reaches scanner evidence normalization and report generation and can affect HTML/PDF security reports generated from scanner exports. A Burp, Qualys, Sonar, or custom scanner export includes request headers, cookies, bearer...

1. Influence: Untrusted input or context reaches the reviewed workflow.

Control gap: Input is treated as more trusted than it should be.

Defensive breakpoint: Classify the source as untrusted and constrain it early.

2. Trust Boundary: The workflow passes data into scanner evidence normalization and report generation.

Control gap: A deterministic policy check is missing or incomplete.

Defensive breakpoint: Validate before the data crosses the boundary.

3. Control Failure: A Burp, Qualys, Sonar, or custom scanner export includes request headers, cookies, bearer tokens, API keys, session identifiers, request bodies, or response snippets. normalize_findings.py copies the raw item...

Control gap: Authorization, validation, isolation, or evidence minimization does not fully hold.

Defensive breakpoint: Add a recursive redaction layer before storing normalized evidence and before rendering reports. Redact key names and values matching Authorization, Cookie, Set-Cookie, token, secret, passw...

4. Impact: The unsafe path can affect HTML/PDF security reports generated from scanner exports.

Control gap: Blast radius is not contained.

Defensive breakpoint: Limit privileges, scope, data exposure, and egress.

5. Detect / Break: Create a fixture with nested request and response objects containing Authorization, Cookie, Set-Cookie, api_key, password, bearer token, private key markers, and session IDs. Run normalize_findings.py and gene...

Control gap: Regression coverage or monitoring may be missing.

Defensive breakpoint: Turn the regression test and detection point into a release gate.

Attacker goal: Influence scanner evidence normalization and report generation so the workflow reaches HTML/PDF security reports generated from scanner exports in an unintended way.

Preconditions: The skill explicitly treats scanner output as untrusted input, and the report agent turns that input into downstream deliverables. In an agentic workflow, this expands the blast radius of sensitive scanner evidence unless redaction is deterministic.

Control breaks: Missing or incomplete deterministic enforcement at the trust boundary.

Impact: HTML/PDF security reports generated from scanner exports

Detection points: Create a fixture with nested request and response objects containing Authorization, Cookie, Set-Cookie, api_key, password, bearer token, private key markers, and session IDs. Run normalize_findings.py and generate_report.py, then assert none of the secret val...

Defensive breakpoints: Add a recursive redaction layer before storing normalized evidence and before rendering reports. Redact key names and values matching Authorization, Cookie, Set-Cookie, token, secret, password, api_key, bearer, private_key, session, credential, client_secret, and configurable organization patterns. Prefer a safe evide...

Safety note: Conceptual defensive flow only. No exploit payloads or unauthorized testing steps are provided.

HB-DET-003

Medium

2. MCP manifest auditor can under-classify nested permissive tool schemas

Confidence: medium

Affected component: MCP tool contract audit

Affected file: scripts/mcp_manifest_audit.py:68-98

Affected endpoint: Not provided

Affected asset: MCP risk tier assignment and tool safety classification

Evidence


```
{
  "summary": "The auditor reads only top-level schema properties for explicit issue generation. It may see nested words through JSON stringification, but it does not recursively report nested free-form parameters, unconstrained object schemas, additionalProperties, arrays of action objects, or weakly constrained nested URL/path/command/body fields.",
  "positive_control": "The example manifest correctly flags a top-level command parameter as T4.",
  "gap": "A tool can hide broad authority under a neutral top-level object such as config, request, options, action, or payload."
}
```


Attack Path

An MCP server exposes a tool with a benign top-level schema but nested command, URL, path, headers, body, credential, or action fields. The current auditor may not clearly emit high-risk nested JSON-path issues, causing the orchestrator to underestimate the server's risk tier.

AI / Agent Relevance

MCP tools are privileged agent boundaries. Under-classifying nested tool authority can make prompt injection or malicious tool metadata more dangerous in a connected agent environment.

Threat Intelligence

{
  "as_of": "2026-05-28",
  "summary": "MCP security guidance emphasizes authorization boundaries, SSRF, token handling, local server risk, and scope minimization. Nested schemas can hide exactly the kind of authority that needs strict review.",
  "sources": [
    {
      "name": "Model Context Protocol Security Best Practices",
      "url": "https://modelcontextprotocol.io/docs/tutorials/security/security_best_practices",
      "relevance": "Covers MCP-specific implementation risks including SSRF, token passthrough, authorization, and local server compromise."
    },
    {
      "name": "OWASP Agentic Skills Top 10",
      "url": "https://owasp.org/www-project-agentic-skills-top-10/",
      "relevance": "Skill and tool execution-layer risks map to dangerous hidden capabilities and weak validation."
    }
  ],
  "signals": [
    {
      "theme": "Nested dangerous capability hiding",
      "relevance": "A neutral top-level object can contain nested command, URL, credential, body, or action fields that change the true risk tier.",
      "recommended_action": "Walk schemas recursively and report exact JSON paths for risky nested fields and permissive object shapes."
    }
  ],
  "mapped_techniques": {
    "mitre_atlas": "Tool abuse via agent capability",
    "cwe": "CWE-20; CWE-749"
  },
  "detection_or_test": "Add nested MCP manifest fixtures with command, URL, headers, body, credential, and additionalProperties paths; assert T4 classification.",
  "freshness": "Checked against official sources on 2026-05-28",
  "confidence": "High"
}

Framework Mappings

{
  "owasp_llm": "Excessive Agency; Insecure Plugin Design",
  "owasp_agentic": "Tool Misuse; Agentic Supply Chain",
  "mitre_atlas": "Tool abuse and execution via agent capability",
  "nist_ai_rmf": "Map, Measure, Manage",
  "cwe": "CWE-20; CWE-749",
  "cve": null
}

Score

```
{
  "system": "AIVSS",
  "value": "Medium",
  "rationale": "The weakness affects detection quality for potentially high-risk tools. Impact depends on the reviewed MCP manifests and whether live high-privilege MCP tools are connected."
}
```


Recommended Fix

Walk input schemas recursively and report exact JSON paths. Flag additionalProperties true or absent on object-like inputs, nested sensitive parameter names, arrays of action objects, missing required fields, broad string parameters, and URL/path/command-like fields without enums, allowlists, regex constraints, or approval language.

Regression Test

Add a manifest fixture with inputSchema.properties.config.properties.command, inputSchema.properties.request.properties.url, nested headers/body fields, and additionalProperties true. Assert the auditor returns T4 and reports each nested JSON path.

Residual Risk

Static manifest checks cannot prove runtime authorization. High-risk MCP tools still need runtime review, allowlists, approval gates, and audit logs.

Attack Path Visualization

For Hades Blackwatch Skill Repository, MCP manifest auditor can under-classify nested permissive tool schemas can be understood as a control failure where untrusted influence reaches MCP tool contract audit and can affect MCP risk tier assignment and tool safety classification. An MCP server exposes a tool with a benign top-level schema but nested command, URL, path, headers, body, credential, or action fields. The...

1. Influence: Untrusted input or context reaches the reviewed workflow.

Control gap: Input is treated as more trusted than it should be.

Defensive breakpoint: Classify the source as untrusted and constrain it early.

2. Trust Boundary: The workflow passes data into MCP tool contract audit.

Control gap: A deterministic policy check is missing or incomplete.

Defensive breakpoint: Validate before the data crosses the boundary.

3. Control Failure: An MCP server exposes a tool with a benign top-level schema but nested command, URL, path, headers, body, credential, or action fields. The current auditor may not clearly emit high-risk nested JSON-path issue...

Control gap: Authorization, validation, isolation, or evidence minimization does not fully hold.

Defensive breakpoint: Walk input schemas recursively and report exact JSON paths. Flag additionalProperties true or absent on object-like inputs, nested sensitive parameter names, arrays of action objects, missi...

4. Impact: The unsafe path can affect MCP risk tier assignment and tool safety classification.

Control gap: Blast radius is not contained.

Defensive breakpoint: Limit privileges, scope, data exposure, and egress.

5. Detect / Break: Add a manifest fixture with inputSchema.properties.config.properties.command, inputSchema.properties.request.properties.url, nested headers/body fields, and additionalProperties true. Assert the auditor return...

Control gap: Regression coverage or monitoring may be missing.

Defensive breakpoint: Turn the regression test and detection point into a release gate.

Attacker goal: Influence MCP tool contract audit so the workflow reaches MCP risk tier assignment and tool safety classification in an unintended way.

Preconditions: MCP tools are privileged agent boundaries. Under-classifying nested tool authority can make prompt injection or malicious tool metadata more dangerous in a connected agent environment.

Control breaks: Missing or incomplete deterministic enforcement at the trust boundary.

Impact: MCP risk tier assignment and tool safety classification

Detection points: Add a manifest fixture with inputSchema.properties.config.properties.command, inputSchema.properties.request.properties.url, nested headers/body fields, and additionalProperties true. Assert the auditor returns T4 and reports each nested JSON path.

Defensive breakpoints: Walk input schemas recursively and report exact JSON paths. Flag additionalProperties true or absent on object-like inputs, nested sensitive parameter names, arrays of action objects, missing required fields, broad string parameters, and URL/path/command-like fields without enums, allowlists, regex constraints, or app...

Safety note: Conceptual defensive flow only. No exploit payloads or unauthorized testing steps are provided.

HB-DET-002

Medium

3. Surface mapper scans generated reports and previous report inputs, polluting later analysis

Confidence: high

Affected component: local attack surface discovery

Affected file: scripts/surface_mapper.py:76-92; .gitignore:9

Affected endpoint: Not provided

Affected asset: review evidence quality and finding accuracy

Evidence


```
{
  "summary": "DEFAULT_SKIP_DIRS excludes common build and dependency folders but not reports/ or
out/. The rerun included generated report HTML, PDF, and report input JSON as if they were source
surfaces. The surface counts increased after report generation: mcp rose to 32, authz to 57, and
secrets to 55 because prior reports contained those words.",
  "observed_noise": [
    "reports/hades-blackwatch-fullstack-review-input.json",
    "reports/hades-blackwatch-skill-repository-hades-blackwatch-full-stack-security-review.html",
    "reports/hades-blackwatch-skill-repository-hades-blackwatch-full-stack-security-review.pdf"
  ],
  "control_gap": "reports/ is ignored in .gitignore but not skipped by surface_mapper.py."
}
```


Attack Path

Generated reports contain security terminology and possibly sensitive evidence. A later full-stack analysis scans those reports and treats them as current code/docs evidence, producing duplicated findings, stale evidence, or false positives.

AI / Agent Relevance

Generated reports can become context for subsequent agent runs. If the tool cannot separate primary source from generated evidence, specialist agents may anchor on stale report content rather than current system behavior.

Threat Intelligence

```
{
  "as_of": "2026-05-28",
  "summary": "Agentic review workflows are vulnerable to stale or attacker-shaped context when generated artifacts are re-ingested as if they were primary source evidence.",
  "sources": [
    {
      "name": "OWASP Top 10 for Agentic Applications",
      "url": "https://genai.owasp.org/2025/12/09/owasp-top-10-for-agentic-applications-the-benchmark-for-agentic-security-in-the-age-of-autonomous-ai/",
      "relevance": "Agent memory, context poisoning, and tool reliability risks map to stale generated evidence being reused by agents."
    },
    {
      "name": "MITRE ATLAS",
      "url": "https://atlas.mitre.org/",
      "relevance": "Provides AI-system threat mapping for poisoning and evasion-adjacent behaviors."
    }
  ],
  "signals": [
    {
      "theme": "Generated artifact context poisoning",
      "relevance": "Reports can contain old findings, keywords, and sensitive context that distort later local scans.",
      "recommended_action": "Exclude reports and other generated outputs by default; require explicit opt-in when auditing report artifacts."
    }
  ],
  "mapped_techniques": {
    "mitre_atlas": "Data poisoning / evidence manipulation analogue",
    "cwe": "CWE-703"
  },
  "detection_or_test": "Add generated report fixtures and assert surface_mapper.py excludes them unless an explicit include-generated option is used.",
  "freshness": "Checked against official sources on 2026-05-28",
  "confidence": "High"
}
```


Framework Mappings


```
{
  "owasp_llm": "Context/Data Poisoning analogue for review evidence",
  "owasp_agentic": "Evidence Integrity / Context Poisoning",
  "mitre_atlas": "Data poisoning and evasion-adjacent evidence manipulation",
  "nist_ai_rmf": "Measure, Manage",
  "cwe": "CWE-703",
  "cve": null
}
```


Score

```
{
  "system": "AIVSS",
  "value": "Medium",
  "rationale": "The issue does not directly execute code or expose new data, but it materially degrades the accuracy and repeatability of agentic security reviews."
}
```


Recommended Fix

Add reports/, out/, coverage/, .pytest_cache/, .mypy_cache/, and other generated-output paths to DEFAULT_SKIP_DIRS. Add --include-generated and --exclude options for controlled overrides. Consider reading .gitignore patterns for default exclusions.

Regression Test

Create reports/generated.html containing token, MCP, credential, and authorization terms. Run surface_mapper.py and assert reports/generated.html is excluded by default. Then run with an explicit include-generated option and assert it appears.

Residual Risk

Some users may intentionally want to audit report artifacts. That should be explicit and separate from source surface mapping.

Attack Path Visualization

For Hades Blackwatch Skill Repository, Surface mapper scans generated reports and previous report inputs, polluting later analysis can be understood as a control failure where untrusted influence reaches local attack surface discovery and can affect review evidence quality and finding accuracy. Generated reports contain security terminology and possibly sensitive evidence. A later full-stack analysis scans those rep...

1. Influence: Untrusted input or context reaches the reviewed workflow.

Control gap: Input is treated as more trusted than it should be.

Defensive breakpoint: Classify the source as untrusted and constrain it early.

2. Trust Boundary: The workflow passes data into local attack surface discovery.

Control gap: A deterministic policy check is missing or incomplete.

Defensive breakpoint: Validate before the data crosses the boundary.

3. Control Failure: Generated reports contain security terminology and possibly sensitive evidence. A later full-stack analysis scans those reports and treats them as current code/docs evidence, producing duplicated findings, sta...

Control gap: Authorization, validation, isolation, or evidence minimization does not fully hold.

Defensive breakpoint: Add reports/, out/, coverage/, .pytest_cache/, .mypy_cache/, and other generated-output paths to DEFAULT_SKIP_DIRS. Add --include-generated and --exclude options for controlled overrides. C...

4. Impact: The unsafe path can affect review evidence quality and finding accuracy.

Control gap: Blast radius is not contained.

Defensive breakpoint: Limit privileges, scope, data exposure, and egress.

5. Detect / Break: Create reports/generated.html containing token, MCP, credential, and authorization terms. Run surface_mapper.py and assert reports/generated.html is excluded by default. Then run with an explicit include-gener...

Control gap: Regression coverage or monitoring may be missing.

Defensive breakpoint: Turn the regression test and detection point into a release gate.

Attacker goal: Influence local attack surface discovery so the workflow reaches review evidence quality and finding accuracy in an unintended way.

Preconditions: Generated reports can become context for subsequent agent runs. If the tool cannot separate primary source from generated evidence, specialist agents may anchor on stale report content rather than current system behavior.

Control breaks: Missing or incomplete deterministic enforcement at the trust boundary.

Impact: review evidence quality and finding accuracy

Detection points: Create reports/generated.html containing token, MCP, credential, and authorization terms. Run surface_mapper.py and assert reports/generated.html is excluded by default. Then run with an explicit include-generated option and assert it appears.

Defensive breakpoints: Add reports/, out/, coverage/, .pytest_cache/, .mypy_cache/, and other generated-output paths to DEFAULT_SKIP_DIRS. Add --include-generated and --exclude options for controlled overrides. Consider reading .gitignore patterns for default exclusions.

Safety note: Conceptual defensive flow only. No exploit payloads or unauthorized testing steps are provided.

HB-DET-005

Low

4. Documentation describes scanner auto-detection before the registry implementation exists

Confidence: high

Affected component: README and documentation accuracy

Affected file: README.md:159; README.md:228; README.md:440; docs/index.html:753; docs/index.html:895

Affected endpoint: Not provided

Affected asset: user expectations for scanner integrations

Evidence


```
{
  "summary": "README and docs state that scanner backends are detected and that the skill chooses the safest integration path. The roadmap also says scanner_registry.py still needs to be added. The current implementation has a scanner-registry agent role but no executable scanner registry script.",
  "current_implementation": "Role guidance exists in agents/roles/scanner-registry.md, but scanner_registry.py is roadmap-only.",
  "impact": "Users may expect automatic detection of Sonar, Burp, Qualys, Semgrep, CodeQL, or Trivy when the current repo mainly supports manual/report-driven workflows."
}
```


Attack Path

A user relies on documentation that implies automated scanner detection, then assumes coverage exists when no scanner backend was actually queried or imported. This can create false assurance in a security review.

AI / Agent Relevance

Agentic tools should be explicit about what evidence was actually collected. Overstated capability descriptions can cause both humans and agents to over-trust missing scanner coverage.

Threat Intelligence

{
  "as_of": "2026-05-28",
  "summary": "Threat-informed security reporting depends on evidence provenance. Overstating scanner or MCP coverage can create false assurance and hide missing controls.",
  "sources": [
    {
      "name": "NIST AI RMF Generative AI Profile",
      "url": "https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence",
      "relevance": "Supports transparent risk measurement, governance, and lifecycle accountability."
    },
    {
      "name": "OWASP Top 10 for Agentic Applications",
      "url": "https://genai.owasp.org/2025/12/09/owasp-top-10-for-agentic-applications-the-benchmark-for-agentic-security-in-the-age-of-autonomous-ai/",
      "relevance": "Agentic tool reliability and overreliance risks are relevant when documentation implies missing scanner coverage."
    }
  ],
  "signals": [
    {
      "theme": "False assurance through capability claims",
      "relevance": "Users may assume Burp, Qualys, Sonar, Semgrep, CodeQL, or Trivy evidence was collected when only local scripts or examples ran.",
      "recommended_action": "Separate implemented, designed, and roadmap capabilities in documentation and final reports."
    }
  ],
  "mapped_techniques": {
    "mitre_atlas": "Defense evasion via false assurance",
    "cwe": "CWE-1059"
  },
  "detection_or_test": "Add a docs consistency check that compares capability claims to implemented scripts and report evidence.",
  "freshness": "Checked against official sources on 2026-05-28",
  "confidence": "Medium"
}

Framework Mappings

{
  "owasp_llm": "Overreliance / Supply Chain Tooling Gap",
  "owasp_agentic": "Tool Reliability / Coverage Gap",
  "mitre_atlas": "Defense evasion via false assurance",
  "nist_ai_rmf": "Govern, Map, Measure",
  "cwe": "CWE-1059",
  "cve": null
}

Score

```
{
  "system": "AIVSS",
  "value": "Low",
  "rationale": "This is a documentation and expectation risk, not a direct vulnerability. It matters because the skill is intended to drive security decisions."
}
```


Recommended Fix

Adjust wording to say Hades Blackwatch is designed to support scanner detection and currently supports local scripts, report ingestion, examples, and role guidance. Keep auto-detection language only after scanner_registry.py exists and is tested.

Regression Test

Add a docs consistency checklist or test that verifies README capability claims match implemented scripts and roadmap status.

Residual Risk

Even after scanner registry implementation, reports should always list which backends actually ran, which were skipped, and why.

Attack Path Visualization

For Hades Blackwatch Skill Repository, Documentation describes scanner auto-detection before the registry implementation exists can be understood as a control failure where untrusted influence reaches README and documentation accuracy and can affect user expectations for scanner integrations. A user relies on documentation that implies automated scanner detection, then assumes coverage exists when no scanner backend...

1. Influence: Untrusted input or context reaches the reviewed workflow.

Control gap: Input is treated as more trusted than it should be.

Defensive breakpoint: Classify the source as untrusted and constrain it early.

2. Trust Boundary: The workflow passes data into README and documentation accuracy.

Control gap: A deterministic policy check is missing or incomplete.

Defensive breakpoint: Validate before the data crosses the boundary.

3. Control Failure: A user relies on documentation that implies automated scanner detection, then assumes coverage exists when no scanner backend was actually queried or imported. This can create false assurance in a security rev...

Control gap: Authorization, validation, isolation, or evidence minimization does not fully hold.

Defensive breakpoint: Adjust wording to say Hades Blackwatch is designed to support scanner detection and currently supports local scripts, report ingestion, examples, and role guidance. Keep auto-detection lang...

4. Impact: The unsafe path can affect user expectations for scanner integrations.

Control gap: Blast radius is not contained.

Defensive breakpoint: Limit privileges, scope, data exposure, and egress.

5. Detect / Break: Add a docs consistency checklist or test that verifies README capability claims match implemented scripts and roadmap status.

Control gap: Regression coverage or monitoring may be missing.

Defensive breakpoint: Turn the regression test and detection point into a release gate.

Attacker goal: Influence README and documentation accuracy so the workflow reaches user expectations for scanner integrations in an unintended way.

Preconditions: Agentic tools should be explicit about what evidence was actually collected. Overstated capability descriptions can cause both humans and agents to over-trust missing scanner coverage.

Control breaks: Missing or incomplete deterministic enforcement at the trust boundary.

Impact: user expectations for scanner integrations

Detection points: Add a docs consistency checklist or test that verifies README capability claims match implemented scripts and roadmap status.

Defensive breakpoints: Adjust wording to say Hades Blackwatch is designed to support scanner detection and currently supports local scripts, report ingestion, examples, and role guidance. Keep auto-detection language only after scanner_registry.py exists and is tested.

Safety note: Conceptual defensive flow only. No exploit payloads or unauthorized testing steps are provided.

HB-DET-004

Low

5. Hades rename changed the normalized finding schema without compatibility notes

Confidence: high

Affected component: finding normalization and downstream interoperability

Affected file: scripts/normalize_findings.py:127

Affected endpoint: Not provided

Affected asset: normalized findings consumed by reports, CI, or downstream tooling

Evidence


```
{
  "summary": "normalize_findings.py now emits schema: hades-blackwatch.findings.v1 after the rename. Existing historical reports, dashboards, or downstream consumers may still expect a previous skill schema, but no compatibility note or alias is documented.",
  "observed_output": "schema: hades-blackwatch.findings.v1",
  "legacy_output": "previous skill schema",
  "expected_output": "schema: hades-blackwatch.findings.v1 with documented migration behavior for older consumers"
}
```


Attack Path

Downstream systems or report consumers route, validate, or store findings based on the schema value. An undocumented schema migration can cause failed validation, incorrect ownership, broken dashboards, or mixed historical data after the skill rename.

AI / Agent Relevance

Agentic review workflows depend on machine-readable handoffs. Undocumented schema changes weaken provenance and can cause downstream agents or integrations to misinterpret the evidence source.

Threat Intelligence

{
  "as_of": "2026-05-28",
  "summary": "Agentic security workflows rely on provenance, source identity, and repeatable machine-readable handoffs. Undocumented schema migrations reduce trust in downstream automation and audit evidence.",
  "sources": [
    {
      "name": "NIST AI RMF Generative AI Profile",
      "url": "https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence",
      "relevance": "Governance and risk management practices emphasize traceability and lifecycle controls."
    },
    {
      "name": "OWASP Agentic Skills Top 10",
      "url": "https://owasp.org/www-project-agentic-skills-top-10/",
      "relevance": "Skill provenance and governance risks apply to reusable agent skill packages."
    }
  ],
  "signals": [
    {
      "theme": "Schema compatibility gap",
      "relevance": "A renamed skill with a new schema can break routing, validation, dashboards, and later agent interpretation if migration behavior is not documented.",
      "recommended_action": "Document aliases or migration behavior and add a schema regression test."
    }
  ],
  "mapped_techniques": {
    "mitre_atlas": "Defense evasion through tooling inconsistency",
    "cwe": "CWE-1104"
  },
  "detection_or_test": "Assert normalize_findings.py emits hades-blackwatch.findings.v1 or a documented compatibility alias.",
  "freshness": "Checked against official sources on 2026-05-28",
  "confidence": "High"
}

Framework Mappings

{
  "owasp_llm": "Supply Chain / Model and Tooling Integrity",
  "owasp_agentic": "Agentic Supply Chain / Tool Contract Drift",
  "mitre_atlas": "Defense evasion through tooling inconsistency",
  "nist_ai_rmf": "Map, Measure, Manage",
  "cwe": "CWE-1104",
  "cve": null
}

Score

```
{
  "system": "AIVSS",
  "value": "Medium",
  "rationale": "The issue affects integration integrity rather than direct exploitability. It can still break automated review pipelines and evidence provenance during migration."
}
```


Recommended Fix

Document the Hades Blackwatch schema in references/report-format.md. If backward compatibility is needed, add schema_aliases or schema_version fields for previous skill schemas and explain how old reports should be handled.

Regression Test

Run normalize_findings.py against examples/scanner-findings.example.json and assert the schema field equals hades-blackwatch.findings.v1.

Residual Risk

Previously generated reports may still contain old schema identifiers. Migration notes should clarify that old reports are historical artifacts.

Attack Path Visualization

For Hades Blackwatch Skill Repository, Hades rename changed the normalized finding schema without compatibility notes can be understood as a control failure where untrusted influence reaches finding normalization and downstream interoperability and can affect normalized findings consumed by reports, CI, or downstream tooling. Downstream systems or report consumers route, validate, or store findings based on the sche...

1. Influence: Untrusted input or context reaches the reviewed workflow.

Control gap: Input is treated as more trusted than it should be.

Defensive breakpoint: Classify the source as untrusted and constrain it early.

2. Trust Boundary: The workflow passes data into finding normalization and downstream interoperability.

Control gap: A deterministic policy check is missing or incomplete.

Defensive breakpoint: Validate before the data crosses the boundary.

3. Control Failure: Downstream systems or report consumers route, validate, or store findings based on the schema value. An undocumented schema migration can cause failed validation, incorrect ownership, broken dashboards, or mix...

Control gap: Authorization, validation, isolation, or evidence minimization does not fully hold.

Defensive breakpoint: Document the Hades Blackwatch schema in references/report-format.md. If backward compatibility is needed, add schema_aliases or schema_version fields for previous skill schemas and explain...

4. Impact: The unsafe path can affect normalized findings consumed by reports, CI, or downstream tooling.

Control gap: Blast radius is not contained.

Defensive breakpoint: Limit privileges, scope, data exposure, and egress.

5. Detect / Break: Run normalize_findings.py against examples/scanner-findings.example.json and assert the schema field equals hades-blackwatch.findings.v1.

Control gap: Regression coverage or monitoring may be missing.

Defensive breakpoint: Turn the regression test and detection point into a release gate.

Attacker goal: Influence finding normalization and downstream interoperability so the workflow reaches normalized findings consumed by reports, CI, or downstream tooling in an unintended way.

Preconditions: Agentic review workflows depend on machine-readable handoffs. Undocumented schema changes weaken provenance and can cause downstream agents or integrations to misinterpret the evidence source.

Control breaks: Missing or incomplete deterministic enforcement at the trust boundary.

Impact: normalized findings consumed by reports, CI, or downstream tooling

Detection points: Run normalize_findings.py against examples/scanner-findings.example.json and assert the schema field equals hades-blackwatch.findings.v1.

Defensive breakpoints: Document the Hades Blackwatch schema in references/report-format.md. If backward compatibility is needed, add schema_aliases or schema_version fields for previous skill schemas and explain how old reports should be handled.

Safety note: Conceptual defensive flow only. No exploit payloads or unauthorized testing steps are provided.

HB-DET-006

Low

6. Smoke checks are manual and not enforced as automated regression tests

Confidence: high

Affected component: test and release assurance

Affected file: scripts/generate_report.py; scripts/normalize_findings.py; scripts/mcp_manifest_audit.py; scripts/surface_mapper.py

Affected endpoint: Not provided

Affected asset: future reliability of security review outputs

Evidence


```
{
  "summary": "Local smoke checks passed for skill validation, surface mapping, MCP manifest audit, prompt pack generation, normalization, and report generation. However, these checks are not codified as a test suite or CI gate.",
  "successful_checks": [
    "quick_validate.py returned Skill is valid",
    "mcp_manifest_audit.py classified run_shell_command as T4",
    "normalize_findings.py produced one normalized finding",
    "generate_report.py produced HTML and PDF outputs"
  ],
  "gap": "No committed tests assert redaction, report generation, schema name, surface exclusion, nested MCP schemas, or PDF/HTML output integrity."
}
```


Attack Path

A future change regresses redaction, report escaping, schema names, surface exclusions, or MCP risk classification. Without automated tests, the regression may be discovered only after a report is shared or a review misses a risky tool.

AI / Agent Relevance

Security skills become reusable agent infrastructure. Regression in the skill can mislead future security reviews and downstream agent decisions.

Threat Intelligence

```
{
  "as_of": "2026-05-28",
  "summary": "Reusable AI security skills become supply-chain components. Automated regression gates are needed so redaction, MCP risk classification, schema identity, and report rendering do not silently weaken over time.",
  "sources": [
    {
      "name": "OWASP Agentic Skills Top 10",
      "url": "https://owasp.org/www-project-agentic-skills-top-10/",
      "relevance": "Reusable skills, plugins, and agent execution layers require governance, validation, and supply-chain controls."
    },
    {
      "name": "NIST AI RMF Generative AI Profile",
      "url": "https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence",
      "relevance": "Supports lifecycle risk management and repeatable controls for GenAI systems."
    }
  ],
  "signals": [
    {
      "theme": "Agentic skill supply-chain regression",
      "relevance": "Security review tooling can become a weak link if its own controls are not tested before integrations expand.",
      "recommended_action": "Create tests for redaction, report output, schema name, surface exclusions, and recursive MCP schema classification; run them in CI."
    }
  ],
  "mapped_techniques": {
    "mitre_atlas": "Defense evasion through tooling weakness",
    "cwe": "CWE-1059"
  },
  "detection_or_test": "Add unittest or pytest coverage plus a CI workflow for the security-critical script paths.",
  "freshness": "Checked against official sources on 2026-05-28",
  "confidence": "High"
}
```


Framework Mappings


```
{
  "owasp_llm": "Supply Chain Vulnerabilities",
  "owasp_agentic": "Agentic Supply Chain / Tool Reliability",
  "mitre_atlas": "Defense evasion through tooling weakness",
  "nist_ai_rmf": "Measure, Manage",
  "cwe": "CWE-1059",
  "cve": null
}
```


Score

```
{
  "system": "AIVSS",
  "value": "Low",
  "rationale": "This is an assurance gap, but it affects security-critical review tooling."
}
```


Recommended Fix

Add a standard-library unittest or pytest suite for redaction, normalization schema, HTML escaping, PDF generation, surface exclusions, and recursive MCP schema risk classification. Add a CI workflow to run skill validation and tests on every change.

Regression Test

Create tests/test_normalize_findings.py, tests/test_generate_report.py, tests/test_mcp_manifest_audit.py, and tests/test_surface_mapper.py with the current examples plus security fixtures.

Residual Risk

Automated tests should be supplemented by manual review for new scanner adapters, new MCP integrations, and report template changes.

Attack Path Visualization

For Hades Blackwatch Skill Repository, Smoke checks are manual and not enforced as automated regression tests can be understood as a control failure where untrusted influence reaches test and release assurance and can affect future reliability of security review outputs. A future change regresses redaction, report escaping, schema names, surface exclusions, or MCP risk classification. Without automated tests, the re...

1. Influence: Untrusted input or context reaches the reviewed workflow.

Control gap: Input is treated as more trusted than it should be.

Defensive breakpoint: Classify the source as untrusted and constrain it early.

2. Trust Boundary: The workflow passes data into test and release assurance.

Control gap: A deterministic policy check is missing or incomplete.

Defensive breakpoint: Validate before the data crosses the boundary.

3. Control Failure: A future change regresses redaction, report escaping, schema names, surface exclusions, or MCP risk classification. Without automated tests, the regression may be discovered only after a report is shared or a...

Control gap: Authorization, validation, isolation, or evidence minimization does not fully hold.

Defensive breakpoint: Add a standard-library unittest or pytest suite for redaction, normalization schema, HTML escaping, PDF generation, surface exclusions, and recursive MCP schema risk classification. Add a C...

4. Impact: The unsafe path can affect future reliability of security review outputs.

Control gap: Blast radius is not contained.

Defensive breakpoint: Limit privileges, scope, data exposure, and egress.

5. Detect / Break: Create tests/test_normalize_findings.py, tests/test_generate_report.py, tests/test_mcp_manifest_audit.py, and tests/test_surface_mapper.py with the current examples plus security fixtures.

Control gap: Regression coverage or monitoring may be missing.

Defensive breakpoint: Turn the regression test and detection point into a release gate.

Attacker goal: Influence test and release assurance so the workflow reaches future reliability of security review outputs in an unintended way.

Preconditions: Security skills become reusable agent infrastructure. Regression in the skill can mislead future security reviews and downstream agent decisions.

Control breaks: Missing or incomplete deterministic enforcement at the trust boundary.

Impact: future reliability of security review outputs

Detection points: Create tests/test_normalize_findings.py, tests/test_generate_report.py, tests/test_mcp_manifest_audit.py, and tests/test_surface_mapper.py with the current examples plus security fixtures.

Defensive breakpoints: Add a standard-library unittest or pytest suite for redaction, normalization schema, HTML escaping, PDF generation, surface exclusions, and recursive MCP schema risk classification. Add a CI workflow to run skill validation and tests on every change.

Safety note: Conceptual defensive flow only. No exploit payloads or unauthorized testing steps are provided.

Personalized Remediation Roadmap

HB-DET-001

Add a recursive redaction layer before storing normalized evidence and before rendering reports. Redact key names and values matching Authorization, Cookie, Set-Cookie, token, secret, password, api_key, bearer, private_key, session, credential, client_secret, and configurable organization patterns. Prefer a safe evidence summary allowlist over storing full raw scanner items by default.

HB-DET-003

Walk input schemas recursively and report exact JSON paths. Flag additionalProperties true or absent on object-like inputs, nested sensitive parameter names, arrays of action objects, missing required fields, broad string parameters, and URL/path/command-like fields without enums, allowlists, regex constraints, or approval language.

HB-DET-002

Add reports/, out/, coverage/, .pytest_cache/, .mypy_cache/, and other generated-output paths to DEFAULT_SKIP_DIRS. Add --include-generated and --exclude options for controlled overrides. Consider reading .gitignore patterns for default exclusions.

HB-DET-005

Adjust wording to say Hades Blackwatch is designed to support scanner detection and currently supports local scripts, report ingestion, examples, and role guidance. Keep auto-detection language only after scanner_registry.py exists and is tested.

HB-DET-004

Document the Hades Blackwatch schema in references/report-format.md. If backward compatibility is needed, add schema_aliases or schema_version fields for previous skill schemas and explain how old reports should be handled.

HB-DET-006

Add a standard-library unittest or pytest suite for redaction, normalization schema, HTML escaping, PDF generation, surface exclusions, and recursive MCP schema risk classification. Add a CI workflow to run skill validation and tests on every change.

Regression Test Plan

HB-DET-001

Create a fixture with nested request and response objects containing Authorization, Cookie, Set-Cookie, api_key, password, bearer token, private key markers, and session IDs. Run normalize_findings.py and generate_report.py, then assert none of the secret values appear in normalized JSON, HTML, or PDF bytes.

HB-DET-003

Add a manifest fixture with inputSchema.properties.config.properties.command, inputSchema.properties.request.properties.url, nested headers/body fields, and additionalProperties true. Assert the auditor returns T4 and reports each nested JSON path.

HB-DET-002

Create reports/generated.html containing token, MCP, credential, and authorization terms. Run surface_mapper.py and assert reports/generated.html is excluded by default. Then run with an explicit include-generated option and assert it appears.

HB-DET-005

Add a docs consistency checklist or test that verifies README capability claims match implemented scripts and roadmap status.

HB-DET-004

Run normalize_findings.py against examples/scanner-findings.example.json and assert the schema field equals hades-blackwatch.findings.v1.

HB-DET-006

Create tests/test_normalize_findings.py, tests/test_generate_report.py, tests/test_mcp_manifest_audit.py, and tests/test_surface_mapper.py with the current examples plus security fixtures.

Residual Risk and Assumptions

Residual risk depends on complete remediation, regression test adoption, scanner coverage, evidence redaction, and continued enforcement of deterministic authorization around model-driven actions.

Overall verdict: Request changes: high-risk issues require remediation before approval.

Report title: Hades Blackwatch Detailed Full-Stack Analysis