



GADRIEL

PRE - PRODUCTION VALIDATION

DEPLOYMENT ASSURANCE

Penetration Test Report

demo.owasp-juice.shop

PREPARED FOR

ABC

PREPARED BY

Gadriel AI Corp.

PRODUCT

Gadriel Deployment Assurance (GDA)

ENGAGEMENT DATE

October 6, 2026

CLASSIFICATION

Confidential

Powered by Prancer SwarmHack v3.3.0

Confidential & Proprietary — Gadriel AI Corp. © 2026. Do not distribute or forward without written consent.



Confidentiality & Notice

Gadriel AI Corp. explicitly disclaims all warranties, whether expressed or implied, with respect to the accuracy, reliability, quality, correctness, or freedom from error or omission of this work product, including any implied warranties of merchantability, suitability for a particular purpose, or non-infringement. The material provided herein is delivered on an “as is” basis, and Gadriel AI Corp. shall not be held liable for any inaccuracies contained within.

This report describes the state of the assessed target during the engagement window stated in Section 2. It is a point-in-time result, not a guarantee that no other vulnerabilities exist. Unless otherwise stated in a master services agreement or a specific project assignment, Gadriel AI Corp. assumes no responsibility or liability, including direct, indirect, incidental, consequential, special, or punitive damages, arising from or connected with the use of, or reliance on, any information contained in this document.

This document does not constitute an endorsement of any companies or products mentioned herein. Copyright © 2026 Gadriel AI Corp. All rights reserved. No portion of this document may be reproduced, copied, or altered without the explicit written authorization of Gadriel AI Corp. Absent express written consent for alternative use, this document is confidential and proprietary information belonging to Gadriel AI Corp. and must not be disseminated to or shared with any third party.

In This Report

1 Executive Summary	What we found, how severe it is, and what to do first.
2 Test Scope & Methodology	What was tested, how Gadriel tests, and how evidence is graded.
3 Target Profile	What the engine learned about the target.
4 Results Summary	Severity distribution and every finding at a glance.
5 Detailed Findings	Seven findings, each with evidence, impact, and the fix.
6 Compliance Mapping Matrix	Findings mapped to eight control frameworks.
7 Remediation Roadmap	Prioritized actions and the path to revalidation.
Appendices	Additional observations, assessment coverage, and about Gadriel.



1 Executive Summary

Gadriel AI Corp. ran Gadriel Deployment Assurance (GDA) against the demo.owasp-juice.shop application on behalf of ABC. GDA is pre-release exploit validation, powered by Prancer's SwarmHack engine. The assessment was black-box and unauthenticated: no credentials and no prior knowledge of the application. The engine completed its run in 7m 55s on October 6, 2026.

7 FINDINGS	2 CRITICAL	6 of 7 PROVEN EVIDENCE	12 CROWN JEWELS CAPTURED
---------------	---------------	---------------------------	--------------------------------

Two findings are Critical and both were Exploited: catalog prices can be overwritten, and orders can be placed over a negative-valued basket. Four findings are High: a SQL injection login bypass, default administrator credentials, unvalidated basket quantities, and a blind SQL injection signal on the category parameter. One finding is Informational. Six of the seven carry proven evidence (three Exploited, three Observed). The seventh, time-based SQL injection, is graded Simulated and is reported at that lower grade.

The findings share one root cause: the application trusts the client. Prices, quantities, and order totals are accepted from the client without server-side authority checks, and the login endpoint accepts injected SQL and default credentials. An unauthenticated client can overwrite catalog prices. Any customer session can place a negative-valued order with no payment step. An attacker with no credentials can sign in as an administrator.

Most significant findings

- **Catalog price tampering (Critical, Exploited).** Any client can overwrite a product price, and the change persists in the catalog for every shopper.
- **Negative-valued order at checkout (Critical, Exploited).** A basket line driven negative checks out into a valid order confirmation, with no payment step before order creation.
- **SQL injection authentication bypass (High, Exploited).** The login endpoint accepts a tautology payload and issues a valid session without credentials.
- **Default administrator credentials (High, Observed).** The REST API accepts a well-known administrator login.
- **Unvalidated basket quantity and blind SQL injection (High).** Negative quantities persist server-side, and the category parameter shows a blind SQL injection signal.

Overall risk rating: Critical. Two exploited business-logic flaws let an attacker change what the application charges, and the authentication layer offers no barrier. We found no compensating controls. Fix the Critical and High findings first, then run revalidation against the fixed build.



2 Test Scope & Methodology

Test scope

Client	ABC
Target	https://demo.owasp-juice.shop/
Assessment type	Pre-release exploit validation: black-box penetration test of a web application and its REST API
Authentication	None (black-box)
API scan	OpenAPI / REST
Engine	SwarmHack v3.3.0 (build v3.3.0-185-ga9cc8319)
Execution mode	Hybrid: full kill chain plus the Authenticated Exploitation Loop (AEL)
Engagement window	2026-10-06 02:45:39 UTC to 2026-10-06 02:53:34 UTC (7m 55s of a 40 minutes budget)
Control frameworks	PCI-DSS 4.0, NIST CSF 2.0, OWASP Top 10 2021, SOC 2, HIPAA, ISO 27001:2022, DORA, NIS2

How Gadriel tests

GDA validates what a build or environment is deployed into, before it ships. It answers one question: can an attacker exploit this, and what do they get? It validates pre-release builds and environments. It does not monitor or protect live production systems. This engagement targeted a demo instance of the application.

The SwarmHack engine is a swarm of specialized agents that plans its own attack path. It runs a six-phase kill chain: port discovery, recon, discovery, exploitation, post-exploitation, and collection. The web lane crawls the application, tests authentication, and runs deep vulnerability scanners that exploit what they find. The network lane works through service-specific attacks. Agents share what they learn through an IntelligenceBus, so a credential or session captured by one agent is available to the others. In hybrid mode, extracted crown jewels (credentials, tokens, and tampered business records) seed the Authenticated Exploitation Loop, which validates, replays, and escalates in bounded iterations.

The engine operates inside an authorization envelope: named targets, a time limit, and a kill switch that halts exploitation if any operation steps outside scope. Proofs are bounded and side-effect-free, and destructive operations are off by default. Every value this assessment tampered with (a basket quantity, two product prices) was restored after the probe.

How evidence is graded

Every finding carries one of four evidence grades. The grades fail closed: an Exploited label without captured target output is downgraded automatically, and Critical severity is reserved for proven evidence.

- **Exploited.** A real round-trip with captured target output.
- **Observed.** A confirmed probe response, without captured output.
- **AttackPathIdentified.** A viable attack path was identified without being exercised. None in this run.
- **Simulated.** Indicative signal only, without captured target output.

A finding reaches this report only if it carries a crown jewel or was genuinely exploited. Fingerprints, skip notices, and enumeration statistics route to the appendices, so nothing is silently deleted.



3 Target Profile

The engine's passive and active discovery produced the following picture of the target.

Host	demo.owasp-juice.shop, served by Heroku (via heroku-router)
Application	OWASP Juice Shop web application with a REST API
Session mechanism	JWT bearer tokens issued by /rest/user/login
API specification	OpenAPI "NextGen B2B API" v2.0.0 (server prefix /b2b/v2), reachable through /api-docs
Unlisted or sensitive paths	/ftp (disclosed by robots.txt), /api-docs, /metrics
Input fields discovered	category, email, id, keyword, name, q, query, search, type, user, username
Response headers of note	X-Content-Type-Options: nosniff; X-Frame-Options: SAMEORIGIN; Access-Control-Allow-Origin: * (wildcard)

The wildcard Access-Control-Allow-Origin header is an observation, not a graded finding. Review it alongside the API access-control fixes in Section 7.

4 Results Summary

Severity distribution

Critical	High	Medium	Low	Informational
2	4	0	0	1

Evidence grades. 3 Exploited, 3 Observed, 0 AttackPathIdentified, 1 Simulated. Six of seven findings carry proven evidence.

Findings at a glance

#	Finding	Severity	Risk	CWE	Evidence
1	Negative-Valued Order Issued at Checkout	Critical	9.5	CWE-840	Exploited
2	Catalog Price Tampering via Unauthorized PUT	Critical	9.5	CWE-840	Exploited
3	SQL Injection Authentication Bypass on REST Login	High	7.1	CWE-89	Exploited
4	Default Credentials Accepted via REST API	High	7.1	CWE-798	Observed
5	Negative Quantity Accepted on Basket Item	High	7.1	CWE-840	Observed
6	Time-Based Blind SQL Injection (Category Parameter)	High	6.8	CWE-89	Simulated
7	Sensitive File Exposed: /robots.txt	Informational	0.9	CWE-200	Observed

Risk is scored 0 to 10 as severity multiplied by confidence. The engine captured 12 crown jewels as proof of impact: credentials, session tokens, an order confirmation, and tampered business records.



5 Detailed Findings

FINDING 01 Negative-Valued Order Issued at Checkout

CRITICAL**Risk** 9.5/10 | **Evidence** Exploited | **CWE** CWE-840 (Business Logic Errors) | **MITRE ATT&CK** T1190 | **OWASP** A04:2021 Insecure Design

AFFECTED ENDPOINT

`POST /rest/basket/1/checkout (over a basket tampered via /api/BasketItems/1)`

DESCRIPTION

Checkout over a basket holding a persisted negative quantity produced a valid order confirmation. The order total is recomputed on the server, but from a client-controlled quantity, so the tampered line flows straight into a financial record. No payment step precedes order creation, which compounds the flaw. A customer-role session was enough to place a negative-valued order.

WHAT WE OBSERVED

GDA first persisted a quantity of -1 on the basket item (PUT /api/BasketItems/1), then issued an empty checkout request (POST /rest/basket/1/checkout). The server returned HTTP 200 and order confirmation 5267-ad7e13d73eba4d53: a completed order over a negative-valued basket. The tampered quantity was restored after the probe.

VERIFICATION

1. Persist a negative quantity with PUT /api/BasketItems/1.
2. Send POST /rest/basket/1/checkout with no request body.
3. Observe the order confirmation returned over the tampered basket: 5267-ad7e13d73eba4d53.

BUSINESS IMPACT

Fraudulent and negative-value orders, manipulated order totals and revenue, and corrupted order and financial records. Because no payment gate precedes order creation, orders can be generated without valid payment.

REMEDIATION

Recompute and validate every order line server-side at checkout. Reject non-positive quantities, verify line totals against stored catalog prices, and require the payment workflow to complete before any order record is created.

**FINDING 02 Catalog Price Tampering via Unauthorized PUT****CRITICAL**

Risk 9.5/10 | **Evidence** Exploited | **CWE** CWE-840 (Business Logic Errors), CWE-915 (Mass Assignment) | **MITRE ATT&CK** T1190 | **OWASP** A04:2021 Insecure Design

AFFECTED ENDPOINT

PUT /api/Products/1

DESCRIPTION

A catalog price was overwritten with a plain PUT request from a non-privileged client, and the tampered price was read back from the catalog. The trust boundary is drawn one field too wide: a business-critical attribute is accepted from a client with no authority to set it. In the live lab round, even an unauthenticated PUT persisted on this target.

WHAT WE OBSERVED

A PUT to /api/Products/1 carrying {"price": 0.01} returned HTTP 200. A follow-up GET read the price back as 0.01 for "Apple Juice (1000ml)" (original price 1.99), confirming the change persisted globally in the catalog. The same technique persisted against /api/Products/2 (1.50 to 0.01). The probe restored the original prices.

VERIFICATION

1. Send PUT /api/Products/1 with {"price": 0.01}. The server returns HTTP 200.
2. Send GET /api/Products/1. The price reads back as 0.01, persisted (original 1.99).

BUSINESS IMPACT

Any client can set arbitrary catalog prices. That enables purchases at near-zero cost, direct financial loss, and loss of catalog integrity across the storefront. The confirmed unauthenticated path removes the need for a session.

REMEDIATION

Restrict price and other mutation fields to privileged roles and validate them against a server-side source of truth. Never accept business-critical attributes from client sessions that lack the matching authority, and reject writes to read-only catalog fields.

**FINDING 03 SQL Injection Authentication Bypass on REST Login****HIGH**

Risk 7.1/10 | Evidence Exploited | CWE CWE-89 (SQL Injection) | MITRE ATT&CK T1078, T1110 | OWASP A03:2021 Injection

AFFECTED ENDPOINT

POST /rest/user/login

DESCRIPTION

The REST login endpoint interpolates credentials into a SQL query. A tautology payload bypassed authentication entirely, and the endpoint issued a valid authenticated session with no legitimate credentials supplied.

WHAT WE OBSERVED

The payload ' or 1=1-- submitted to /rest/user/login was accepted, and the server issued a session token. The token value is withheld from this report and held in the raw engine evidence.

VERIFICATION

1. Submit the SQL tautology payload ' or 1=1-- to /rest/user/login.
2. Observe that authentication succeeds and a session token is issued without valid credentials.

BUSINESS IMPACT

Complete authentication bypass. An attacker can sign in without credentials, including to administrator accounts, leading to account takeover and unrestricted data access. An injectable login query also puts the underlying database contents at risk.

REMEDIATION

Use parameterized queries or prepared statements for every credential check and database interaction. Add input validation and a WAF ruleset targeting SQL injection as defense in depth.

Control mappings: see Section 6.

**FINDING 04 Default Credentials Accepted via REST API****HIGH**

Risk 7.1/10 | **Evidence** Observed | **CWE** CWE-798 (Use of Hard-coded Credentials) | **MITRE ATT&CK** T1078, T1110 | **OWASP** A07:2021 Identification and Authentication Failures

AFFECTED ENDPOINT`POST /rest/user/login`**DESCRIPTION**

The REST authentication endpoint accepted well-known default administrative credentials and issued an authenticated session. Privileged access required no secret knowledge.

WHAT WE OBSERVED

Login with username admin@juice-sh.op and password admin123 was accepted, and an authenticated bearer-token session was established. The token value is withheld from this report and held in the raw engine evidence.

VERIFICATION

1. Send POST /rest/user/login with admin@juice-sh.op and admin123.
2. Observe that an authenticated administrator session is issued.

BUSINESS IMPACT

Trivial administrative account takeover. Combined with the exposed API surface, an attacker gains control over application data and configuration.

REMEDIATION

Rotate the credential immediately, enforce a strong password policy, and add rate limiting, per-account lockout, and multi-factor authentication to the login endpoint.

Control mappings: see Section 6.

**FINDING 05 Negative Quantity Accepted on Basket Item****HIGH**

Risk 7.1/10 | Evidence Observed | CWE CWE-840 (Business Logic Errors) | MITRE ATT&CK T1190 | OWASP A04:2021 Insecure Design

AFFECTED ENDPOINT

PUT /api/BasketItems/1

DESCRIPTION

The quantity field on a basket item was set to -1, and the server accepted and persisted the value. Nothing server-side bounds the quantity, so a customer can drive an order line negative. This is the root-cause weakness behind the negative-valued checkout in Finding 1.

WHAT WE OBSERVED

PUT /api/BasketItems/1 with {"quantity": -1} returned HTTP 200, and the response reflected the persisted negative quantity. The probe also tested a quantity of 0 and a string type-confusion payload; neither was accepted, so only the sign is unvalidated. The test ran against the test user's own basket, and the original value was restored.

VERIFICATION

1. Send PUT /api/BasketItems/1 with the body {"quantity": -1}.
2. Observe the HTTP 200 response echoing the persisted negative quantity.

BUSINESS IMPACT

Negative line items enable order-total manipulation, abuse of refunds and credits, and corrupted inventory and financial records.

REMEDIATION

Validate quantity server-side as a positive integer bounded above by available stock, and reject it before any total is computed. Never accept a client-controlled sign on a quantity or amount field.

**FINDING 06 Time-Based Blind SQL Injection (Category Parameter)****HIGH**

Risk 6.8/10 | Evidence Simulated | CWE CWE-89 (SQL Injection) | MITRE ATT&CK T1190 | OWASP A03:2021 Injection

AFFECTED ENDPOINT

GET /?category=1' OR '1'='1

DESCRIPTION

A response-time differential on the category parameter is consistent with time-based blind SQL injection. The evidence grade is Simulated: the engine saw the signal but did not carry out automated data extraction, so this finding is reported with the lower grade and should be confirmed during revalidation.

WHAT WE OBSERVED

With the payload 1' OR '1'='1, the response took 9,371 ms against an 853 ms baseline, a delay of about 6.8 seconds. A second payload, " OR ""=", produced a length and status differential consistent with boolean-based blind injection on the same parameter. The delay appeared with a boolean tautology rather than an explicit time-delay payload, so timing alone is not conclusive; the boolean differential is the stronger signal. The crawler also discovered these input fields, which should be reviewed for the same weakness: category, email, id, keyword, name, q, query, search, type, user, and username.

VERIFICATION

1. Send GET /?category=1' OR '1'='1.
2. Baseline response time: 853 ms.
3. Payload response time: 9,371 ms (delay of about 6,775 ms).

BUSINESS IMPACT

Blind SQL injection can be used for database enumeration and data exfiltration without triggering error-based detection. With the login bypass in Finding 3 confirmed, the database is likely reachable.

REMEDIATION

Use parameterized queries or prepared statements for all database access, apply strict input validation to the fields listed above, and deploy a WAF with SQL injection rulesets. Confirm closure during revalidation with an explicit time-delay payload and a data-extraction proof.

Control mappings: see Section 6.

**FINDING 07 Sensitive File Exposed: /robots.txt****INFORMATIONAL**

Risk 0.9/10 | **Evidence** Observed | **CWE** CWE-200 (Exposure of Sensitive Information) | **MITRE ATT&CK** T1083 | **OWASP** A01:2021
Broken Access Control

AFFECTED ENDPOINT

GET /robots.txt

DESCRIPTION

The publicly accessible robots.txt file discloses a hidden path through its Disallow directive. robots.txt exists to guide crawlers, but its contents help an attacker map the application's attack surface.

WHAT WE OBSERVED

GET /robots.txt returned HTTP 200 (28 bytes) and disclosed the hidden path /ftp through a Disallow rule.

VERIFICATION

1. Send GET /robots.txt.
2. Observe the disclosed path: Disallow: /ftp.

BUSINESS IMPACT

Minor information disclosure that aids reconnaissance by pointing to a non-public directory.

REMEDIATION

Do not treat robots.txt as a security control. Enforce server-side access control on /ftp and any other sensitive path whether or not it is listed, and remove unnecessary entries.

Control mappings: see Section 6.



6 Compliance Mapping Matrix

GDA mapped findings against 8 control frameworks. Of 22 control requirements tested, 18 are implicated by at least one finding. Each row below is a requirement; the last column lists the findings that implicate it.

Framework	Requirement	Control	Findings
PCI-DSS 4.0	6.2.4	Software engineering techniques prevent injection attacks	3, 6
	8.3.1	All user access to systems is authenticated	4
	6.5.3	Insecure communications and information disclosure	7
NIST CSF 2.0	PR.DS-01	Data-at-rest and data-in-transit protection	3, 6, 7
	PR.AA-01	Identities and credentials are issued and managed	4
OWASP Top 10 2021	A03:2021	Injection	3, 6
	A07:2021	Identification and Authentication Failures	4
	A01:2021	Broken Access Control	7
SOC 2	CC6.1	Logical and physical access controls	3, 4, 6, 7
HIPAA	164.312(a)(1)	Access control	3, 6, 7
	164.312(d)	Person or entity authentication	4
ISO 27001:2022	A.8.26	Application security requirements	3, 6
	A.8.5	Secure authentication	4
	A.8.3	Information access restriction	7
DORA	Art.9(2)	ICT risk management framework	3, 6
	Art.9(3)(a)	Access management policies	4, 7
NIS2	Art.21(2)(e)	Security in network and information systems acquisition, development, maintenance	3, 6
	Art.21(2)(d)	Supply chain security	4, 7

The engine emitted framework mappings for Findings 3, 4, 6, and 7. It emitted none for the business-logic findings (1, 2, and 5). Their OWASP references (A04:2021 Insecure Design) are assigned by CWE and shown in each finding. This engagement also satisfies the universal testing requirements PCI-DSS 4.0 §11.4.1, NIST CSF 2.0 DE.CM-08, SOC 2 CC4.1, DORA Art.26(1) (threat-led penetration testing), and NIS2 Art.21(2)(e).



7 Remediation Roadmap

Priority	Action	Addresses
P1 Immediate	Enforce server-side authority and validation on business-critical fields. Reject non-positive quantities, validate prices and order totals against a trusted catalog, require completed payment before order creation, and restrict mutations to privileged roles.	Findings 1, 2, 5; verb surface
P1 Immediate	Eliminate SQL injection. Convert the login and category queries to parameterized statements, and add a WAF ruleset as defense in depth.	Findings 3, 6
P1 Immediate	Remove default credentials. Rotate the administrator credential, enforce a strong password policy, and add MFA, rate limiting, and per-account lockout to the login endpoint.	Finding 4; no lockout
P1 Immediate	Validate the unconfirmed NoSQL operator reflection on /b2b/v2/orders.	Appendix A
P2 Near term	Harden sessions. Add an expiry (exp) claim to issued JWTs and implement server-side token revocation.	Non-expiring tokens
P2 Near term	Reduce exposed surface. Require authentication for /api-docs, restrict /metrics to internal networks, constrain HTTP verbs to those each endpoint declares, and review the wildcard CORS header.	Appendix A; Section 3
P3 Hardening	Protect /ftp and other sensitive paths regardless of robots.txt, and add SAST and DAST to the development lifecycle.	Finding 7
Verify	Re-run GDA against the fixed build. A finding closes only on a clean revalidation.	All findings

Fix, then revalidate

GDA closes the loop from exploit to fix. Each finding above carries the request that proved it, so your engineers can reproduce it, fix it, and confirm the fix. When the fixes ship, Gadriel re-runs the assessment against the fixed build and reports which findings are closed and which remain.

GDA also runs as a continuous subscription: the same assessment on every release, so a regression surfaces before it ships. That is exploit-to-fix on every release, not once a year. Ask your Gadriel representative about continuous coverage for demo.owasp-juice.shop and the rest of your application portfolio.

Next steps

1. Begin remediation of the two Critical findings immediately, starting with server-side validation of prices, quantities, and totals.
2. Fix the login endpoint (SQL injection and default credentials) in the same release.
3. Schedule revalidation once the fixes are deployed to the pre-release build.



Appendix A Additional Observations

The engine routed the following items to its intelligence-context block. They did not meet the bar for a published finding, but each is real attack surface. Review them, and validate the first one during revalidation.

Possible NoSQL operator injection

[/b2b/v2/orders](#) Unconfirmed · engine severity Critical · confidence 70%

The operator object "\$ne" was reflected in the response for the JSON body parameter cid, while the benign baseline was not. The operator survived into a query layer that echoes it. The engine did not exploit this, so it is not a published finding. Validate it during revalidation, and treat it as a priority because of the engine's Critical severity rating.

API documentation exposed anonymously

[/api-docs](#) Medium · confidence 90%

Interactive Swagger/OpenAPI documentation ("NextGen B2B API" v2.0.0, server prefix /b2b/v2) is reachable without authentication. It maps the API surface for an attacker, including undocumented endpoints.

Observability endpoint exposed

[/metrics](#) Medium · confidence 90%

A Prometheus-format metrics endpoint answered HTTP 200 anonymously. Metrics and debug routes frequently leak internal state and versions, and sometimes personal data.

HTTP method surface beyond the declared API

[/api/Addressss](#), [/api/BasketItems](#), [/api/Cards](#) Medium · confidence 70%

OPTIONS responses advertise PUT, PATCH, and DELETE on endpoints whose discovered API surface declares only GET. The handler accepts verbs the API contract never offered. This is the verb-tampering foothold that the price and quantity findings exploit.

Session tokens carry no expiry

[/rest/user/login](#) Medium · confidence 95%

The JWTs issued at login contain no exp claim (3 of 3 sampled). Without an expiry, a stolen token stays valid for as long as the signing key and any server-side session registry allow (CWE-613).

No account lockout

[/rest/user/login](#) Medium · confidence 90%

A valid login was accepted immediately after 10 consecutive failed attempts on a sacrificial test account, with no per-account lockout or failed-attempt counter (CWE-307). Online password guessing is limited only by any network-level rate limiter.

Appendix B Assessment Coverage

GDA ran its full kill chain and tried to reach well beyond the web tier. The target exposes only an HTTPS web application and REST API, so agents built for other service types found no applicable service and were skipped. That is the expected result for this target, and it records that the engine looked for additional exposed surface and found none:

- Identity and directory attacks (LDAP and SMB enumeration, Kerberos, DCSync, delegation abuse, NTLM relay): no directory service was exposed.



- Host and network services (SSH, SNMP, SMTP, DNS, NTP, TFTP, Telnet, WinRM, IPMI, UPnP, IPP): no such service was reachable on the target.
- Database protocols (MySQL, PostgreSQL, MongoDB, MSSQL): no database service was exposed.
- Container and cloud pivots (Docker API, Kubelet, SSH-based AWS pivot): no applicable service was discovered.
- MCP endpoint discovery (/mcp, /v1/mcp, /api/mcp, /mcp/a, /mcp/b, /sse): no Model Context Protocol surface was found at the probed paths.

The Active Directory certificate (ADCS) and NTLM-downgrade probes were also skipped because they require authenticated domain credentials, which were out of scope for a black-box engagement.

Appendix C About Gadriel

Gadriel AI Corp. is the AI System Assurance company. Our three products validate what AI writes (AI Code Assurance, GCA), what AI does (AI Behavioral Assurance, GBA), and what it is deployed into (Deployment Assurance, GDA), all before it ships. Gadriel validates before release. It does not provide runtime protection or in-flight monitoring.

This report is a GDA deliverable. GDA is powered by Prancer's SwarmHack engine and delivered under the Gadriel brand.