

SECURITY & QA AUDIT REPORT

Ghaymah Infrastructure & CLI v2 Assessment

Lead Auditor: Ziad Mahmoud Ahmed Abdelgwad

Role: Cybersecurity Specialist

Target: Ghaymah CLI v2 (Linux-amd64) & Web Platform

Classification: Confidential / Final Report with Remediations

1. CLI Security Vulnerabilities (MITRE ATT&CK)

Threat ID	Vulnerability	Severity
T1552	Sensitive Information Disclosure (Dockerfile & .env Leak)	Critical
T1539	Session Revocation Failure (Logout Bypass)	High
T1552.001	Unsecured Credentials in Build Servers	Critical

1.1 Sensitive Information Disclosure (T1552)

Description: The auto-generated Dockerfile for static sites aggressively copies the entire root directory (COPY ./) into the public web space. This exposes highly sensitive configuration files, such as .env, to the public internet.

Mitigation: Modify the CLI auto-generation logic to explicitly generate a .dockerignore file excluding .env, .git, and secret files. Alternatively, use specific copy commands (e.g., COPY src/ /app/) rather than wildcard copying.

Proof of Concept (Execution):

```
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# echo "DB_PASSWORD=SuperSecretAdminPassword123" > .env
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# ../gy-linux-amd64 deploy
  Detected: Existing Dockerfile
  Deploy complete!
```

Evidence (Screenshots):

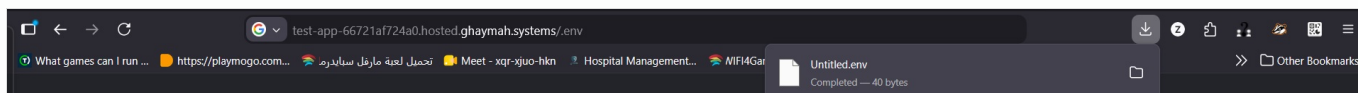


Fig 1. Downloading the .env file containing database passwords directly from the deployed URL.

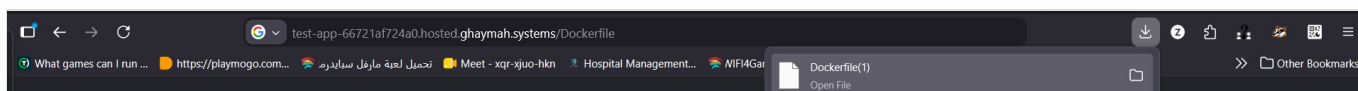


Fig 2. Source code and Dockerfile exposure via browser download.

1.2 Session Revocation Failure (T1539)

Description: The logout command only clears local configuration files but fails to revoke the session token server-side, returning an API error.

Mitigation: Implement a robust backend endpoint (e.g., /api/auth/logout) that actively invalidates the JWT or removes the session record from the database, ensuring the token cannot be reused if intercepted.

```
DESKTOP-QVCBK08:~/ghaymah-v2-test# ./gy-linux-amd64 logout
time=2026-08-19T13:01:32.549Z level=WARN msg="sign out request failed, proceeding with local cleanup"
component=nhost-client component=auth-client error="failed to decode response: error in ReadBody: request
failed with status 400: {\"status\":400,\"message\":\"\\"refreshToken\\" is required\", \"error\": \"invalid-
request\"}"
    Logged out. See you soon!
```

1.3 Unsecured Credentials Storage (T1552.001)

Description: Build server logs reveal that Docker registry credentials are saved without encryption on the build environment, posing a major privilege escalation risk.

Mitigation: Integrate Docker credential helpers (e.g., pass or secretservice) on the build nodes, or pass registry credentials dynamically as ephemeral environment variables during the CI/CD pipeline rather than saving them to config.json.

```
2026-08-19T13:46:28.888903836Z WARNING! Using --password via the CLI is insecure.
2026-08-19T13:46:29.365624526Z WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
2026-08-19T13:46:29.365647119Z Login Succeeded
```

2. Architecture & Backend Reliability Issues

2.1 Configuration Poisoning Leading to CI/CD Hang (DoS)

Description: The platform's configuration synchronization (gy config set env) lacks robust input sanitization. Injecting malformed shell structures, unclosed quotes, or shell substitutions (e.g., `\$(id)`) causes a fatal syntax error within the backend build worker.

Because the backend script evaluates these variables without a try/catch block, the exception is unhandled. The pipeline never reports a "Failed" state back to the GraphQL API, resulting in a permanent Zombie Pipeline (CrashLoopBackOff) that blocks further deployments for the tenant (Self-inflicted DoS).

Mitigation:

- **Input Sanitization:** Validate regex patterns for environment variables before syncing (allow only standard keys).
- **Safe Evaluation:** Do not evaluate user variables directly in shell scripts during the Docker build process.
- **Timeout & Error Handling:** Implement strict timeouts for build processes and catch exceptions to gracefully return a "Failed" status.

```
DESKTOP-QVCBK08:~/ghaymah-v2-test# ../gy-linux-amd64 config set env "RCE_TEST=\$(id)"
DESKTOP-QVCBK08:~/ghaymah-v2-test# ../gy-linux-amd64 deploy
  Deploying test-app...
  Waiting for deployment to complete (this can take a few minutes)...
# [Process hangs indefinitely due to unhandled backend exception]
```

2.2 Deployment State Synchronization Failure (Port Caching)

Description: The CLI fails to detect port changes in updated custom Dockerfiles. It caches older configurations (e.g., Port 80) even when the developer explicitly sets EXPOSE 8080, leading to immediate application crashes and 502 errors.

Mitigation: Force the CLI to re-parse the Dockerfile upon every deployment execution, explicitly clearing the local port cache if a discrepancy is detected between the cached config and the actual file.

```
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# cat Dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY index.html .
EXPOSE 8080
CMD ["python", "-m", "http.server", "8080"]

DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# ../gy-linux-amd64 deploy
  Using existing config: test-app (project: test-app)
  Detected: Existing Dockerfile
  Using configured port: 80
```

Evidence (Screenshots):

```

💡 Tunnel:      gy tunnel start <name>
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# cat << 'EOF' > Dockerfile
index.html .
EX> FROM python:3.9-slim
> WORKDIR /app
http.server", "8080" COPY index.html .
> EXPOSE 8080
> CMD ["python", "-m", "http.server", "8080"]
> EOF
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# cat Dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY index.html .
EXPOSE 8080
CMD ["python", "-m", "http.server", "8080"]
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# ../gy-linux-amd64 deploy
[i] Using existing config: test-app (project: test-app)
[i] Detected: Existing Dockerfile
[i] Using configured port: 80
[i] Using existing project: test-app
[i] Building deployment artifact...
[i] Uploading...
[✓] Uploaded successfully
[i] Deploying test-app...
[i] Waiting for deployment to complete (this can take a few minutes)...
[✓] Deploy complete!

URL:      https://test-app-66721af724a0.hosted.ghaymah.systems
App:      test-app
Project:  test-app

💡 View logs:  gy logs
💡 Edit config: gy config
💡 Tunnel:      gy tunnel start <name>
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app# cat Dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY index.html .
EXPOSE 8080
CMD ["python", "-m", "http.server", "8080"]
DESKTOP-QVCBK08:~/ghaymah-v2-test/test-app#

```

Fig 3. CLI forcing the cached port 80 despite the Dockerfile mapping to 8080.

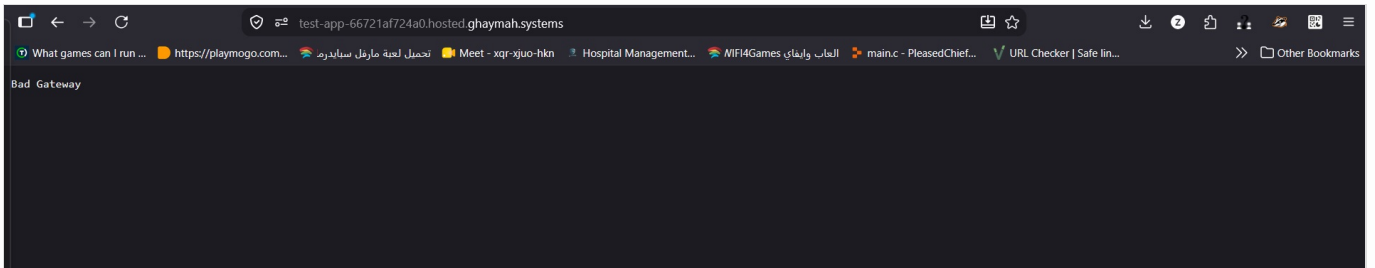


Fig 4. The resulting 502 Bad Gateway error due to mismatched port forwarding.

3. Quality Assurance (QA) Findings

3.1 "Tunnel" Feature is Non-Functional (Ghost Code)

Description: The tunnel start command is present in the CLI but contains no functional networking logic. Debug analysis shows that it successfully authenticates the user, prints a malformed placeholder URL (https://), and exits immediately without establishing any socket connection or reverse proxy.

Mitigation: Hide or remove the command from the CLI binary until the backend tunnel infrastructure is fully implemented to prevent user confusion and false expectations.

```
DESKTOP-QVCBK08:~/ghaymah-v2-test# ./gy-linux-amd64 tunnel start my-tunnel --port 9090
  Starting tunnel: my-tunnel → http://localhost:9090
  Tunnel is live at: https://
DESKTOP-QVCBK08:~/ghaymah-v2-test# [Process exits immediately]
```

4. Web Platform & API Security (Out of CLI Scope)

This section documents critical vulnerabilities discovered in the broader Ghaymah web platform ecosystem, independent of the CLI architecture.

4.1 Broken Authentication & Session Management

CWE-287: Improper Authentication / CWE-620: Unverified Password Change

During the evaluation of the platform's authentication flows, two critical logic flaws were identified in the password management module:

- **Missing OTP Validation on Password Reset:** When a user requests a password reset, the system sends a direct link without requiring an OTP (One-Time Password) or secondary verification. Clicking the reset link immediately logs the user into the active session without forcing a login challenge, presenting a severe Account Takeover (ATO) risk.
- **Unverified Active Password Change:** Once authenticated inside the dashboard, a user can arbitrarily change the account password without being prompted to enter the current (old) password. This violates standard security controls and allows a malicious actor with temporary access (e.g., an unlocked device or a hijacked session) to permanently lock the legitimate user out.

Risk Vector	Impact	Remediation (Mitigation)
Account Takeover (ATO) via Link Interception	Critical. Full compromise of deployed infrastructure.	Implement OTP verification and force manual login post-reset.
Session Hijacking Escalation	High. Lockout of legitimate account owner.	Enforce a "Current Password" field for any credential modification.