Threat Model & Architecture Report (Bedrock)

Generated: 2025-08-12 23:17 UTC

Source: FAISS vectors + Amazon Bedrock (Claude).

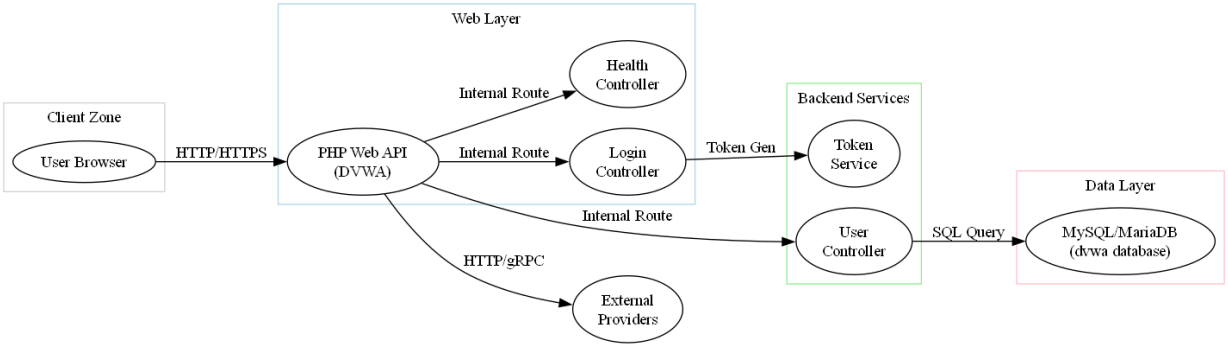
Context Excerpts (vector search)

openapi: 3.0.0
info:
title: 'DVWA API'
contact:
url: 'https://github.com/digininja/DVWA/'
email: robin@digi.ninja
version: '0.1'
servers:
-
url: 'http://dvwa.test'
description: 'API server'
paths:
/vulnerabilities/api/v2/health/echo:
post:
tags:
- health
description: 'Echo, echo, cho, cho, o o ....'
operationId: echo
requestBody:
description: 'Your words.'
content:
application/json:
schema:
$ref: '#/components/schemas/Words'
responses:
'200':
description: 'Successful operation.'
/vulnerabilities/api/v2/health/connectivity:
post:
tags:
- health
description: 'The server occasionally loses connectivity to other systems and so this can be used to check connectivity status.'
operationId: checkConnectivity
requestBody:
description: 'Remote host.'
content:
application/json:

schema:
$ref: '#/components/schemas/Target'
responses:
'200':
description: 'Successful operation.'
/vulnerabilities/api/v2/health/status:
get:
tags:
[file: C:\Users\A1pha.000\Desktop\pentest\defcon 33\defcon33_llm_appsec\repo\vulnerabilities\api\openapi.yml, chunk: 0]

```
<?php

# Start the app with:
#
# php -S localhost:8000 -t public

namespace Src;

use OpenApi\Attributes as OAT;

class HealthController
{
private $command = null;
private $requestMethod = "GET";

public function __construct($requestMethod, $version, $command) {
$this->requestMethod = $requestMethod;
$this->command = $command;
}

#[OAT\Post(
tags: ["health"],
path: '/vulnerabilities/api/v2/health/echo',
operationId: 'echo',
description: 'Echo, echo, cho, cho, o o ....',
parameters: [
new OAT\RequestBody (
description: 'Your words.',
content: new OAT\MediaType(
mediaType: 'ap
```

Architecture Diagram


Client Zone
User Browser
HTTP/HTTPS
Web Layer
PHP Web API
(DVWA)
Internal Route
Health
Controller
Internal Route
Login
Controller
Internal Route
HTTP/gRPC
External
Providers
Token Gen
Backend Services
Token
Service
User
Controller
SQL Query
Data Layer
MySQL/MariaDB
(dvwa database)

STRIDE Threat Model

Overall Risk: high

The DVWA (Damn Vulnerable Web Application) API and application expose multiple security vulnerabilities across authentication, input validation, and token management, designed intentionally as a learning platform for web application security testing.

Authentication	Spoofing, Elevation of Privilege	high	src/Login.php, src/LoginController.php	• Use strong, randomly generated token secrets • Implement proper token validation with cryptographically secure methods • Add token expiration and rotation mechanisms
API Endpoints	Information Disclosure, Tampering	medium	vulnerabilities/api/openapi.yml, src/HealthController.php	• Implement strict input validation • Add authentication checks for all endpoints • Use parameterized queries to prevent injection
Database Configuration	Elevation of Privilege	high	README.md database setup instructions	• Use strong, unique database credentials • Limit database user privileges • Disable authentication only in controlled test environments
Token Management	Spoofing, Information Disclosure	high	src/Login.php with hardcoded token secrets	• Use cryptographically secure random token generation • Implement proper token validation • Never hardcode token secrets