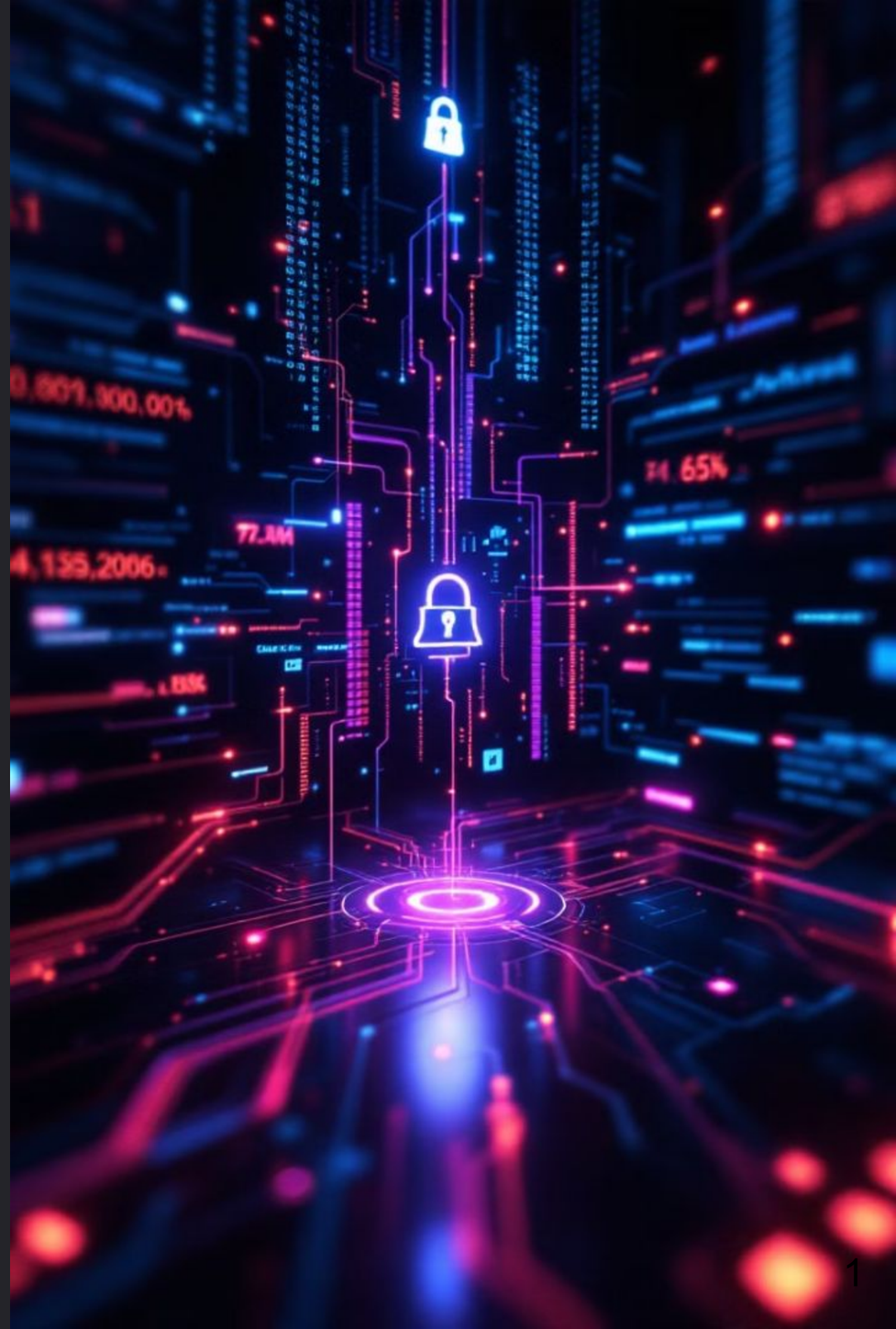
1

# GraphQL Attack Simulation on DVGA - **WASP**

Welcome to our comprehensive GraphQL attack simulation using the Damn Vulnerable GraphQL Application (DVGA).

Prepared by aiegoo tonyleekorea for the KISEC Project 1 on April 9, 2025. I'll be using industry-standard tools including Docker, OWASP ZAP, Burp Suite, and Metasploit to demonstrate real-world GraphQL security vulnerabilities and exploitation techniques.

a **by aiegoo**



# Lab Setup Rules

### Isolate Lab Environment

Never connect to public internet

### Use Trusted Network

Only trusted local connections

### Virtual Environment

Deploy in controlled VM Docker container

### Snapshot Protection

Use VM snapshots for recovery

Clone repo: github.com/dolevf/Damn-Vulnerable-GraphQL-Application

**Project Repo: github.com/aiegoo/kisec-present**

Report deployed at**:** **https://aiegoo.github.io/kisec-present/**

**DVGA:** https://4e4f-14-38-189-145.ngrok-free.app/graphql

**Payload: https://gist.github.com/aiegoo/2ac371f792996dccd321f9fff93a52b9p**



Made with Gamma


# Attack Playbook Overview

### Reconnaissance

Gather intelligence on the target application

### Vulnerability Scanning

Identify potential security weaknesses

### Exploitation

Execute attacks on discovered vulnerabilities

### Reporting

Document findings and recommendations

Our target is the **Damn Vulnerable GraphQL Application (DVGA)**, an intentionally insecure implementation of GraphQL designed for security testing and education. The application will be hosted locally at http://localhost:5013 for our attack simulation.



Made with Gamma

4

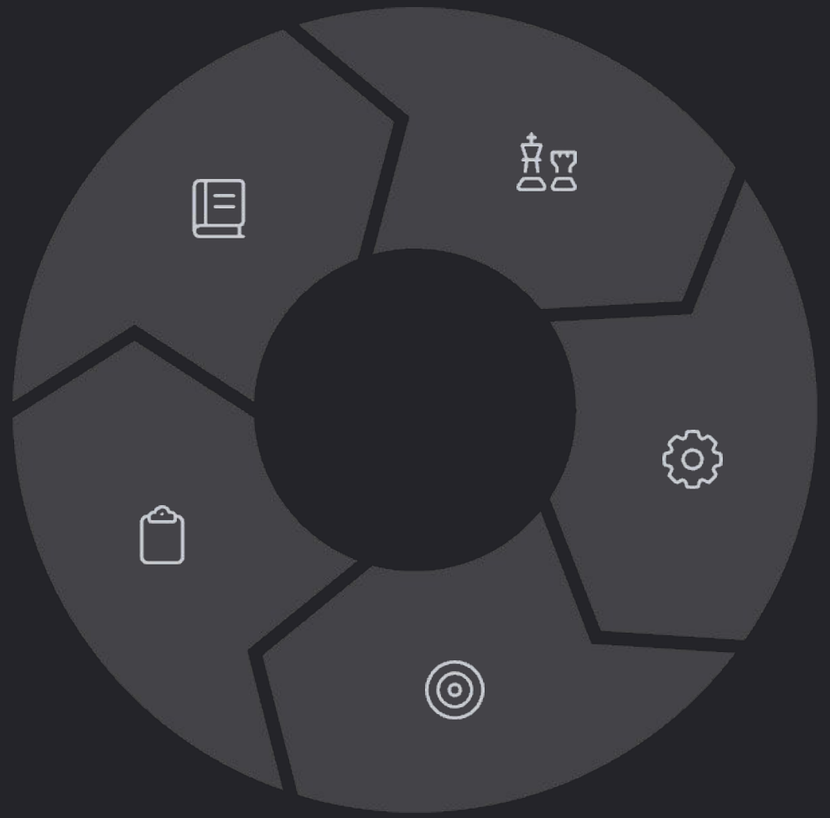
# Project Methodology

## Research

Study GraphQL vulnerabilities and attack vectors

## Playbook Creation

Document step-by-step attack procedures



## Scenario Development

Create realistic attack scenarios

## Tool Selection

Choose appropriate penetration testing tools

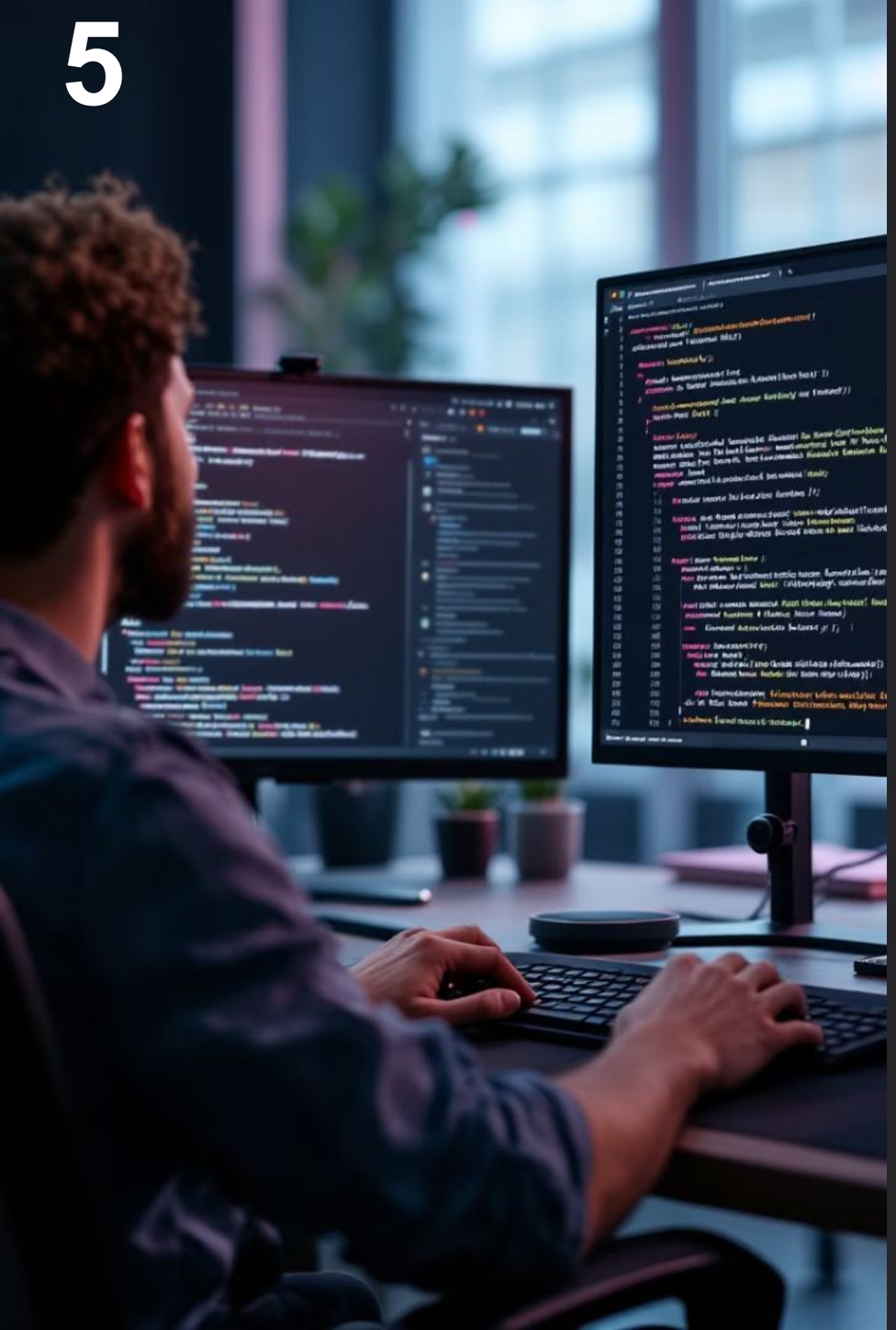
## Target Selection

Configure DVGA as our test environment

Our methodology follows a structured approach to ensure comprehensive testing of all potential GraphQL vulnerabilities. We'll leverage both automated tools and manual techniques to identify and exploit security issues in the target application.



Made with Gamma



# Environment Setup

## Clone Repository

Download the DVGA codebase to your local environment using Git:

```
git clone https://github.com/dolevf/Damn-Vulnerable-GraphQL-Application
```

## Navigate to Directory

Change to the application directory to access configuration files:

```
cd Damn-Vulnerable-GraphQL-Application
```

## Deploy with Docker

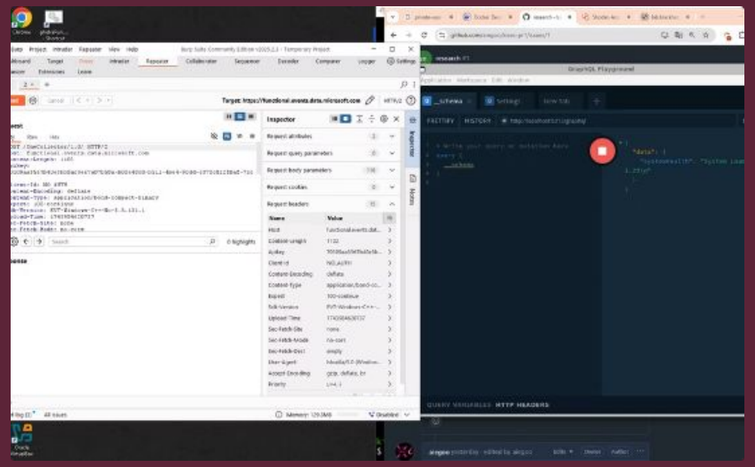
Launch the containerized application with all dependencies:

```
docker-compose up -d
```

Once deployed, verify the application is running by accessing http://localhost:5013 in your browser. The environment isolation provided by Docker ensures consistent testing conditions and prevents interference with other systems.

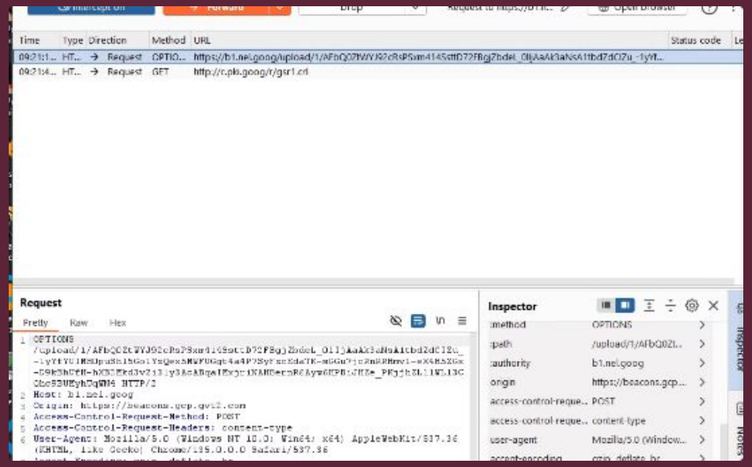
Made with Gamma

# Burp Suite Configuration



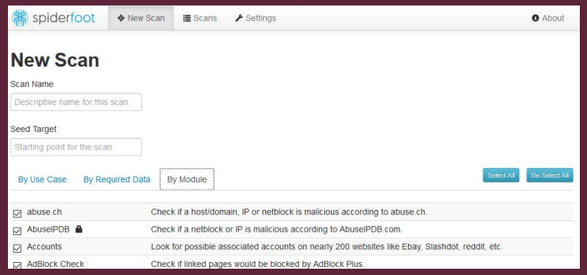
## Proxy Settings

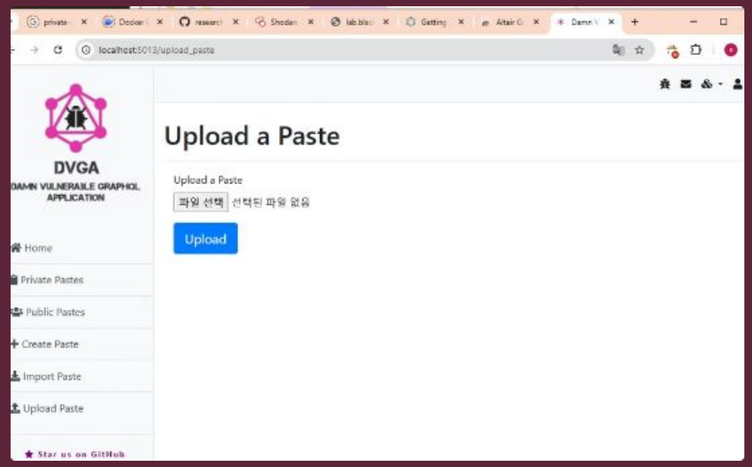
Configure HTTP history and intercept rules




## Request Analyzer

Inspect and modify GraphQL queries




## DVGA Interface

Vulnerable GraphQL testing environment



Made with Gamma

# Reconnaissance Phase

## Network Mapping

Perform service detection on the target network to identify running services and their versions: **nmap -sV -T4 [127.0.0.1]**

## Shodan Intelligence

Search for exposed GraphQL endpoints on public IPs using specialized search operators. Initialize with your API key and run targeted queries: **shodan search graphql**

## Spiderfoot

Launch comprehensive OSINT gathering with web interface at http://localhost:5001

During reconnaissance, we focus on gathering intelligence about our target without actively engaging with it. This passive information collection helps identify potential entry points for later exploitation.



Made with Gamma

8

# Vulnerability Scanning

## Nmap Security Scanning

While primarily a network scanner, Nmap's scripting engine can detect web vulnerabilities:

```
nmap -sV -sC -p- 127.0.0.1
```

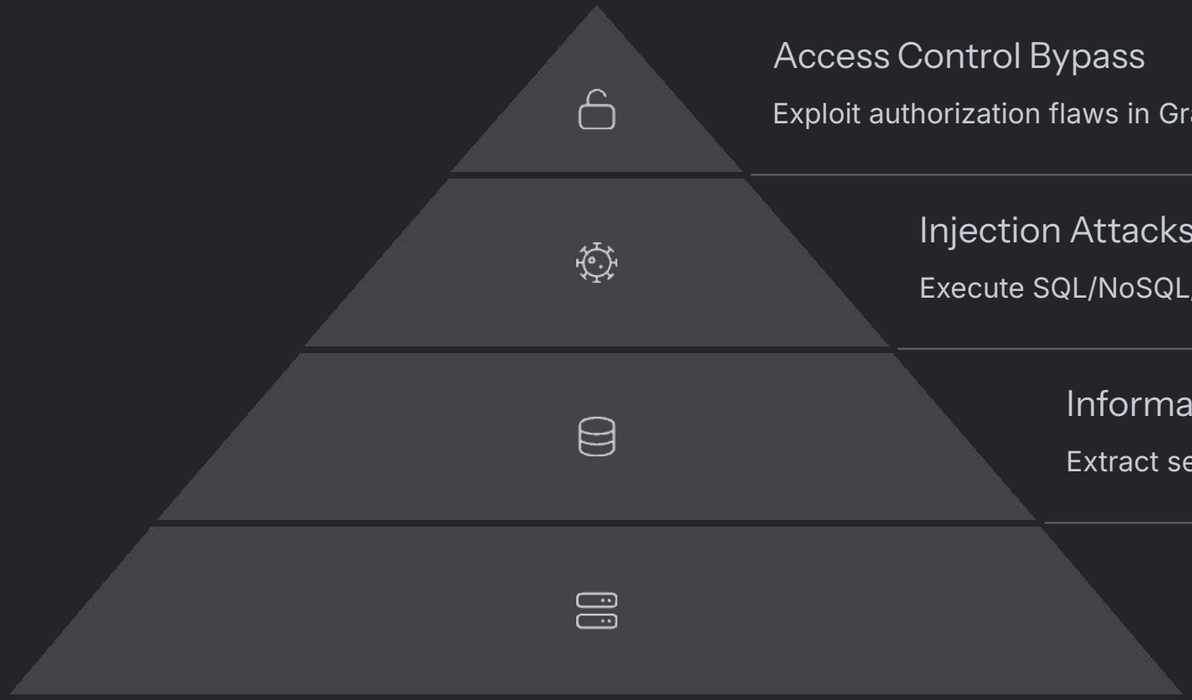
The comprehensive port scan with default scripts (-sC) will identify open ports, running services, and potential security misconfigurations in the application stack supporting GraphQL. Look for unexpected open ports that might indicate backdoors or administrative interfaces with weaker security controls.

Automated vulnerability scanning provides a systematic approach to identifying potential security weaknesses. These tools help prioritize further manual testing efforts on the most promising attack vectors.



Made with Gamma

# Exploitation Phase



### Access Control Bypass

Exploit authorization flaws in GraphQL resolvers

### Injection Attacks

Execute SQL/NoSQL/OS commands through unvalidated inputs

### Information Disclosure

Extract sensitive data through introspection and error messages

### DoS Vulnerabilities

Exploit recursive queries and expensive operations

For exploitation, we'll use two primary tools:

| | |
|---|---|
| **Metasploit Framework** | Launch with **msfconsole** and search for GraphQL-specific modules. When direct modules aren't available, adapt web exploitation techniques to target the GraphQL endpoint. |
| **Burp Suite** | Intercept GraphQL queries and mutations to modify parameters, bypass authentication, exploit IDOR vulnerabilities, and extract sensitive data through introspection. |

During exploitation, document each successful attack vector, including the exact payload used, the vulnerability exploited, and the impact achieved.



Made with Gamma


# Monitoring and Detection

## Log Collection

Configure Wazuh agents to gather logs from application servers, API gateways, and network devices during the attack simulation.

## Rule Configuration

Implement custom detection rules specifically designed to identify GraphQL attack patterns, including malicious queries and abnormal access patterns.

## Real-time Analysis

Monitor detection events during exploitation to evaluate effectiveness of security controls and identify blind spots in monitoring coverage.

## Alert Validation

Verify that significant attack techniques trigger appropriate alerts and evaluate alert fidelity to reduce false positives.

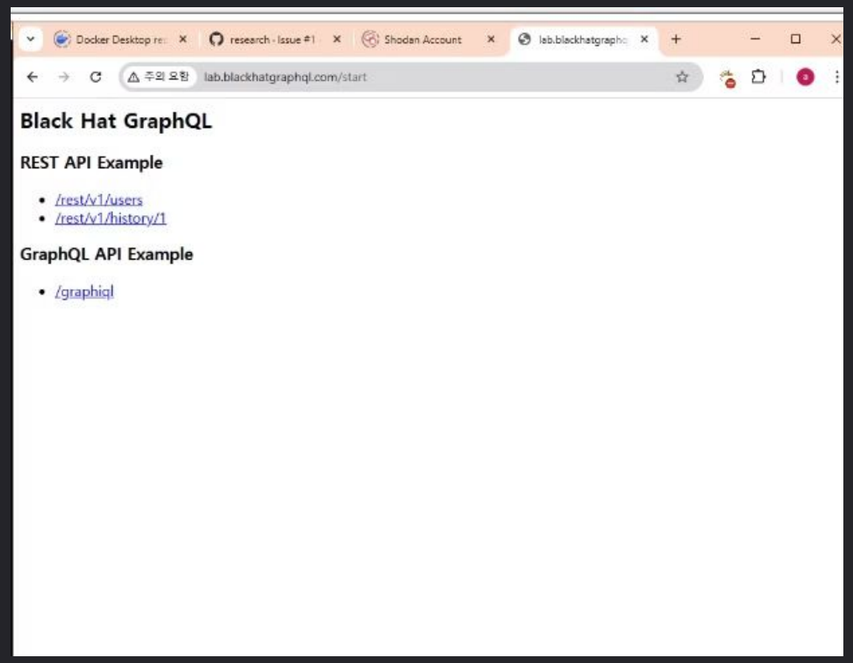
Effective detection capabilities are crucial for defending against GraphQL attacks. By implementing and testing monitoring solutions during our attack simulation, we can verify their effectiveness and identify opportunities for improvement in real-world environments.

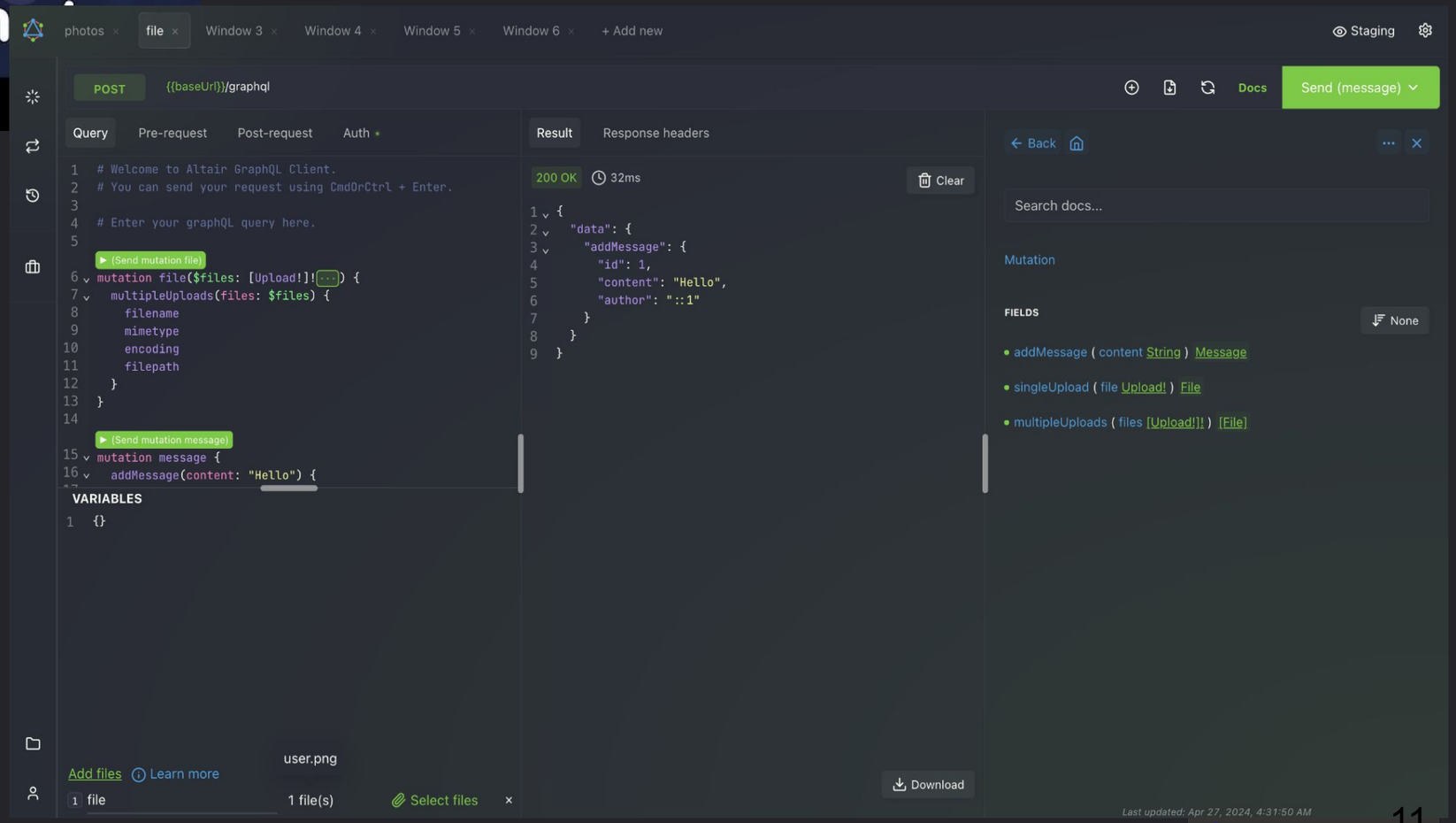


Made with Gamma

# Learning curve

graphql vs REST



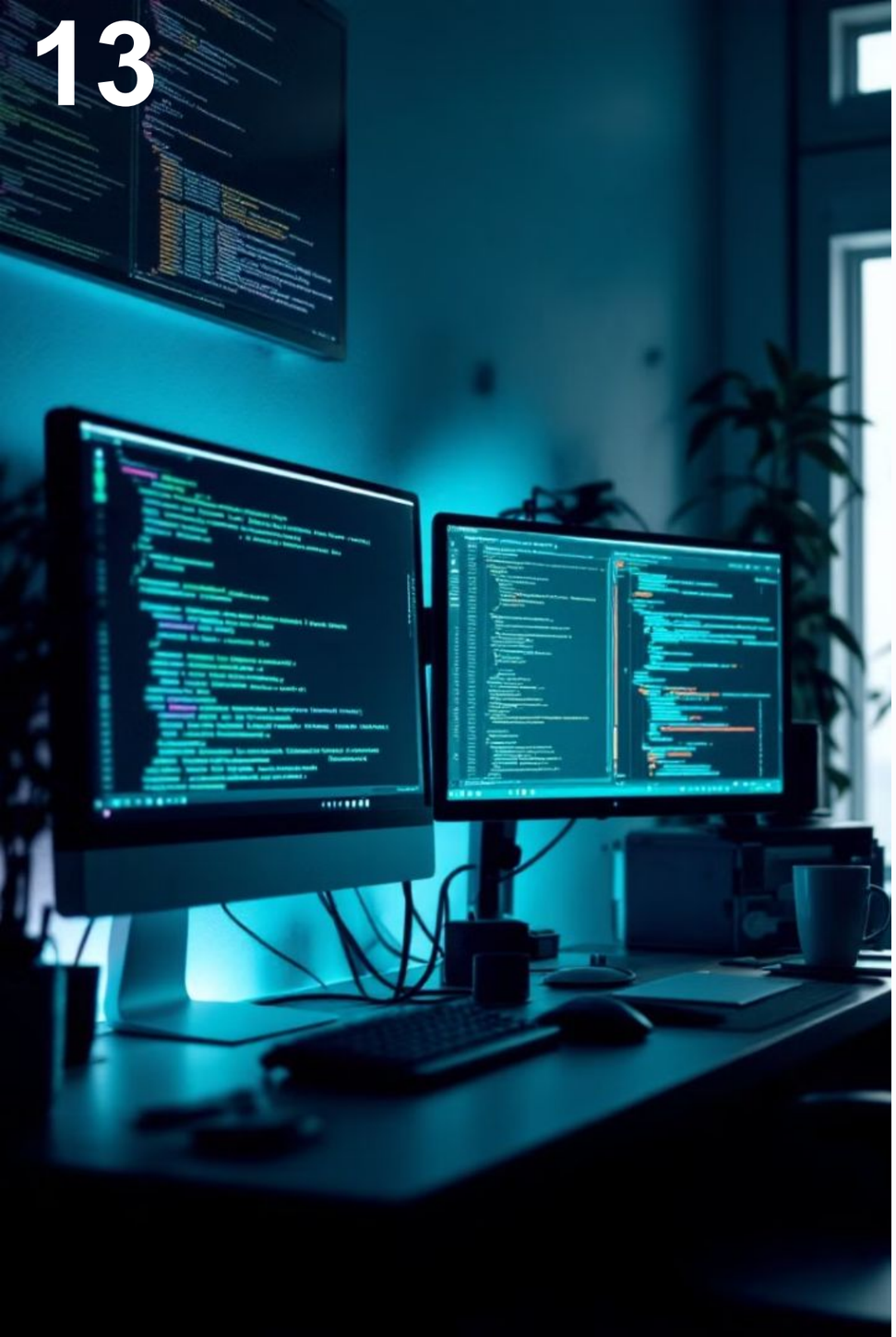

Made with Gamma

# GraphQL Penetration Testing Lab

Setup guide and attack methodology for ethical hackers



Made with Gamma



# Tools Arsenal

## ffuf

Used for endpoint fuzzing to discover GraphQL services and hidden resources.

## curl

Employed for manual introspection queries and direct API interaction.

## Burp Suite

Leveraged for request interception, modification, and GraphQL query analysis.

## Introspection Queries using Altair

Applied to extract schema information and map available operations.

Made with Gamma

# Initial Discovery

## GraphQL Endpoint Located

Successfully identified endpoint at /graphql, providing direct API access.

## Introspection Enabled

Self-documentation feature active, allowing complete schema extraction.

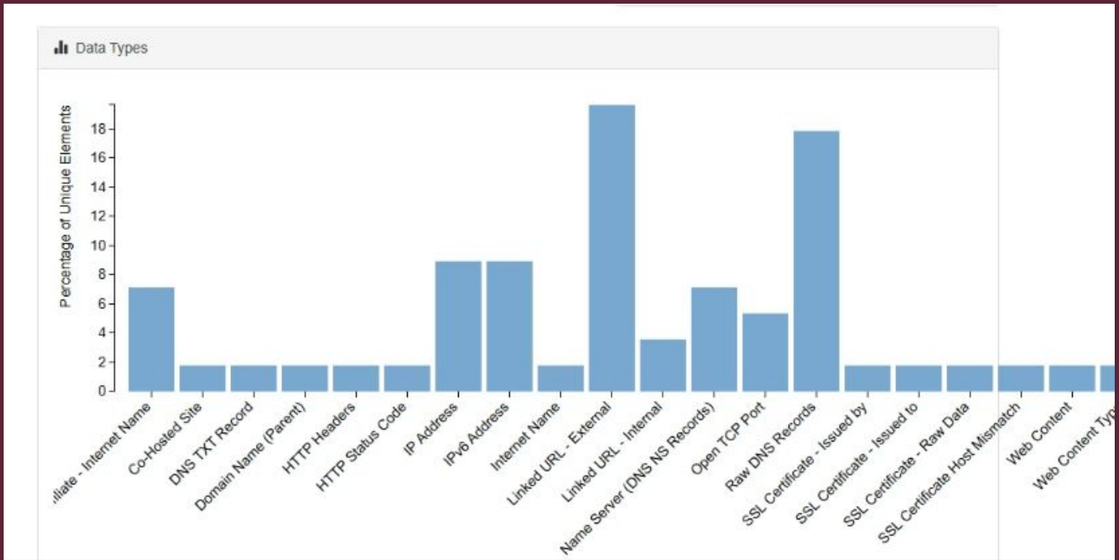
## Schema Extracted

Full API structure obtained, revealing all available operations and types.



Made with Gamma

# Reconnaissance Results




Spiderfoot scan revealed significant exposure vectors

## Critical Findings

- External links exposed
- DNS records enumerated
- SSL certificates reveal tech stack
- Headers offer fingerprinting

DVGA_tH1 RUNNING

Summary | Correlations | Browse | Graph | Scan Settings | Log | Search...

| Type | Unique Data Elements | Total Data Elements | Last Data Element |
|---|---|---|---|
| Co-Hosted Site | 1 | 2 | 2025-04-07 12:57:34 |
| Domain Name (Parent) | 1 | 1 | 2025-04-07 12:57:15 |
| HTTP Headers | 1 | 1 | 2025-04-07 12:57:16 |
| HTTP Status Code | 1 | 1 | 2025-04-07 12:57:16 |
| IP Address | 5 | 5 | 2025-04-07 12:57:15 |
| IPv6 Address | 5 | 5 | 2025-04-07 12:57:15 |
| Internet Name | 153 | 153 | 2025-04-07 12:58:15 |
| Linked URL - External | 11 | 11 | 2025-04-07 12:57:16 |
| Linked URL - Internal | 2 | 2 | 2025-04-07 12:57:16 |
| Open TCP Port | 1 | 1 | 2025-04-07 12:57:34 |
| SSL Certificate - Issued by | 1 | 1 | 2025-04-07 12:57:34 |
| SSL Certificate - Issued to | 1 | 1 | 2025-04-07 12:57:34 |
| SSL Certificate - Raw Data | 1 | 1 | 2025-04-07 12:57:34 |
| SSL Certificate Host Mismatch | 1 | 1 | 2025-04-07 12:57:34 |
| Web Content | 1 | 1 | 2025-04-07 12:57:16 |
| Web Content Type | 1 | 1 | 2025-04-07 12:57:16 |

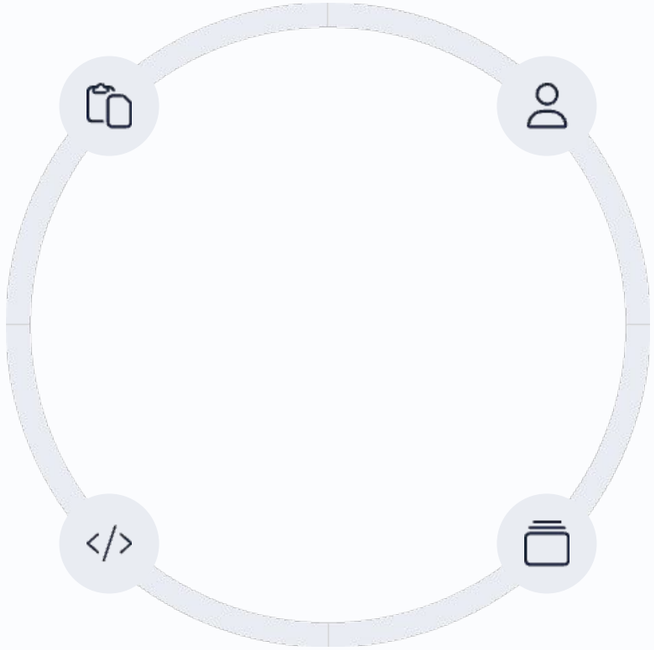
# Schema Components

## PasteObject

User-created content container with potential data exposure risks.

## Mutations

createUser, login, and createPaste operations discovered with no observed auth.

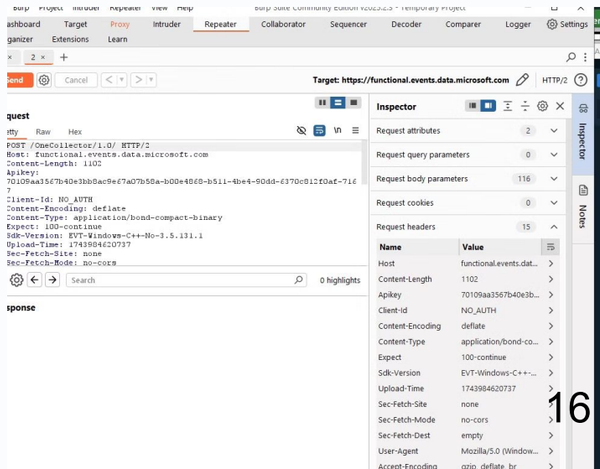


## UserObject

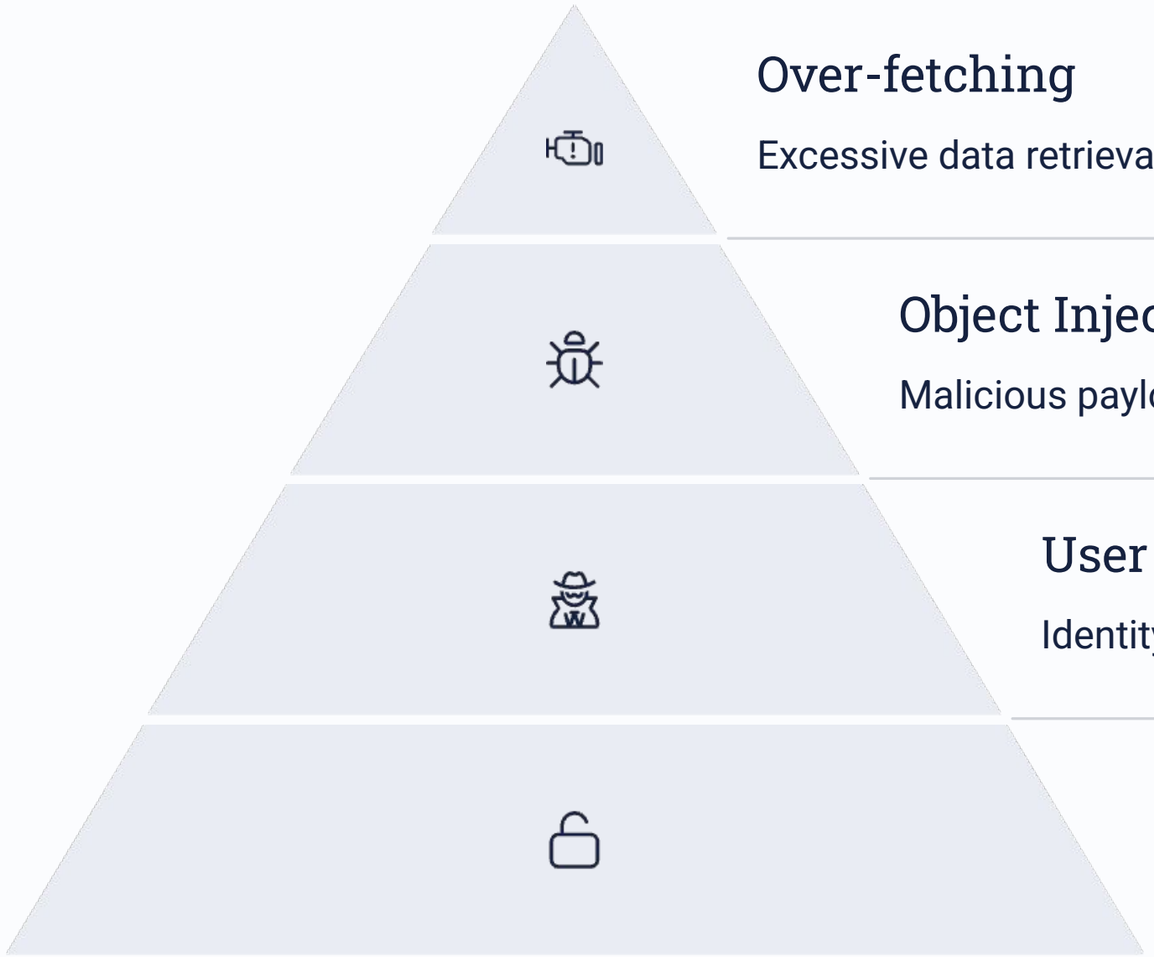
Account information structure exposing user data management operations.

## AuditObject

System activity tracking with possible information disclosure vectors.



# Vulnerabilities Overview



## Over-fetching

Excessive data retrieval potential

## Object Injection

Malicious payloads via inputs

## User Enumeration

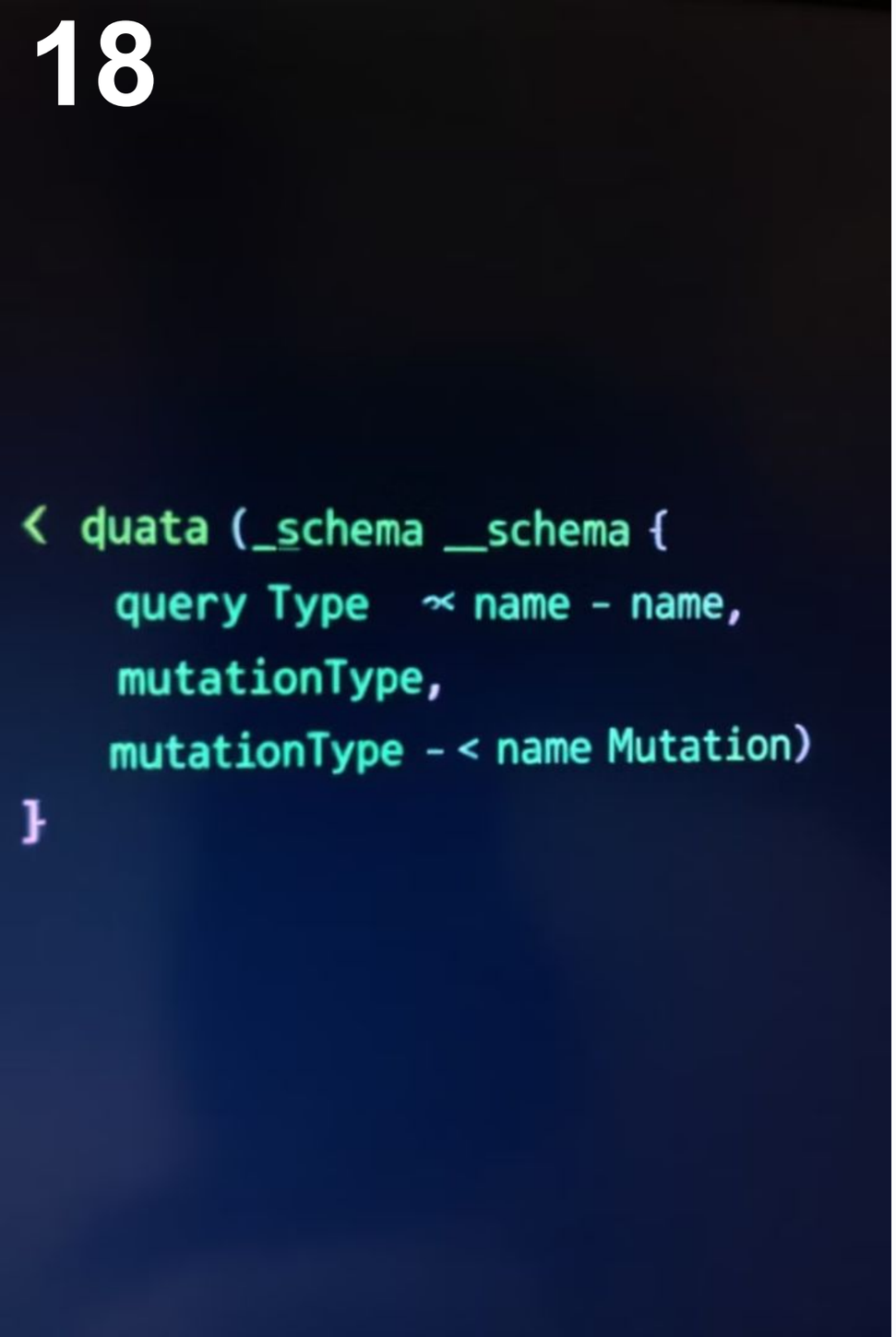
Identity discovery capability

## Root Issues

No auth, exposed schema, no rate limits



Made with Gamma




# Introspection Evidence

### Send Introspection Query

Crafted and dispatched a request for complete schema details.

### Receive Detailed Response

Server returned full schema including types, queries, and mutations.

### Extract Security Implications

Identified sensitive operations and data structures for potential exploitation.

Made with Gamma

# Advanced Reconnaissance

## Nmap Scanning

- Werkzeug HTTP server detected on port 5013
- GraphQL endpoint responds to introspection
- Endpoint validated even without query

## Endpoint Analysis

- Confirmed /graphql endpoint
- Successfully extracted type information
- Discovered Query, Mutation interfaces

## WebSocket Discovery

- HTTP 101 Switching Protocols supported
- /subscriptions endpoint confirmed
- WebSocket security headers identified

# Attack Surface Summary

| Component | Details | Security Implications |
|---|---|---|
| Service | Werkzeug (Python HTTP server) | Known for minimal security defaults |
| Port | 5013/tcp | Exposed network interface |
| Endpoint | /graphql | Primary attack surface |
| Features | Introspection + WebSocket | Expanded attack vectors |

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: aRnlpG8Xwz
```

```
GET /subscriptions HTTP/1.
Host: 0.0.0.0:5013
Connection: Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
Origin: http://localhost:5
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: MV5U83G
```

# WebSocket Inspection

## HTTP Protocol

Initial connection via standard HTTP

## Upgrade Request

Connection upgraded to WebSocket

## Persistent Connection

Allows real-time data exchange

## Introspection Query

Schema discovery via __schema query

### 🔁 HTTP vs WebSocket in GraphQL

| Feature | HTTP (Query/Mutation) | WebSocket (Subscription) |
|---|---|---|
| Protocol | Stateless (request/response) | Persistent (bi-directional) |
| Use Cases | Fetch data or update server | Get real-time updates from server |
| Connection | One-time, closed after response | Long-lived, stays open |
| Efficiency | More overhead for frequent updates | Low latency, ideal for live data |



Made with Gamma

```
kisec@DESKTOP-1C4EIQM: /mnt/c/Users/kisec/Desktop/repos/kisec/vmware/kisec-pr1
Reading state information... Done
The following NEW packages will be installed:
  libdbus-glib-1-2
0 upgraded, 1 newly installed, 0 to remove and 22 not upgraded.
Need to get 65.4 kB of archives.
After this operation, 217 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libdbus-glib-1-2 am
Fetched 65.4 kB in 1s (54.8 kB/s)
Selecting previously unselected package libdbus-glib-1-2:amd64.
(Reading database ... 58878 files and directories currently installed.)
Preparing to unpack .../libdbus-glib-1-2_0.112-2build1_amd64.deb ...
Unpacking libdbus-glib-1-2:amd64 (0.112-2build1) ...
Setting up libdbus-glib-1-2:amd64 (0.112-2build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.9) ...
kisec@DESKTOP-1C4EIQM:/mnt/c/Users/kisec/Desktop/repos/kisec/vmware/kisec-p

desktop-file-install is missing. Skipping /tmp/.mount_graphqwmMG7y/AppRun.
[08:28:36.353] [info] sending to open window SettingsRequest
[08:28:36.358] [info] settings null
[08:44:55.418] [info] sending to focused window Tab Settings
```

```
Protect endpoints w/ IP Intelligence: https://ngrok.com/r/ipintel

Session Status                online
Account                       xraiegoo@gmail.com (Plan: Free)
Version                       3.22.0
Region                        Japan (jp)
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://bd17-182-208-176-28.ngrok-free.app -> http://localhost:5013

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

# External Attack Surface

### Spiderfoot

Automate OSINT gathering

### theHarvester

Email and subdomain enumeration

### Shodan

Discover exposed GraphQL endpoints



Made with Gamma

# Ngrok Tunnel Setup

### Create Tunnel

Expose local server to internet

### Secure Connection

TLS encryption with unique URL

### Command Line

Monitor requests in real-time

```
kisec@DESKTOP-1C4EIQM:/mnt/c/Users/kisec/Desktop/repos/kisec/vmware/kisec-pr1$ curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | \
> sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null &&
> echo "deb https://ngrok-agent.s3.amazonaws.com buster main" | \
> sudo tee /etc/apt/sources.list.d/ngrok.list &&
> sudo apt update && sudo apt install ngrok
[sudo] password for kisec:
deb https://ngrok-agent.s3.amazonaws.com buster main
Get:1 https://ngrok-agent.s3.amazonaws.com buster InRelease [20.3 kB]
Get:2 https://ngrok-agent.s3.amazonaws.com buster/main amd64 Packages [7615 B]
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Fetched 412 kB in 3s (149 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
21 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  ngrok
0 upgraded, 1 newly installed, 0 to remove and 21 not upgraded.
```

# Exposure Analysis

## External Links

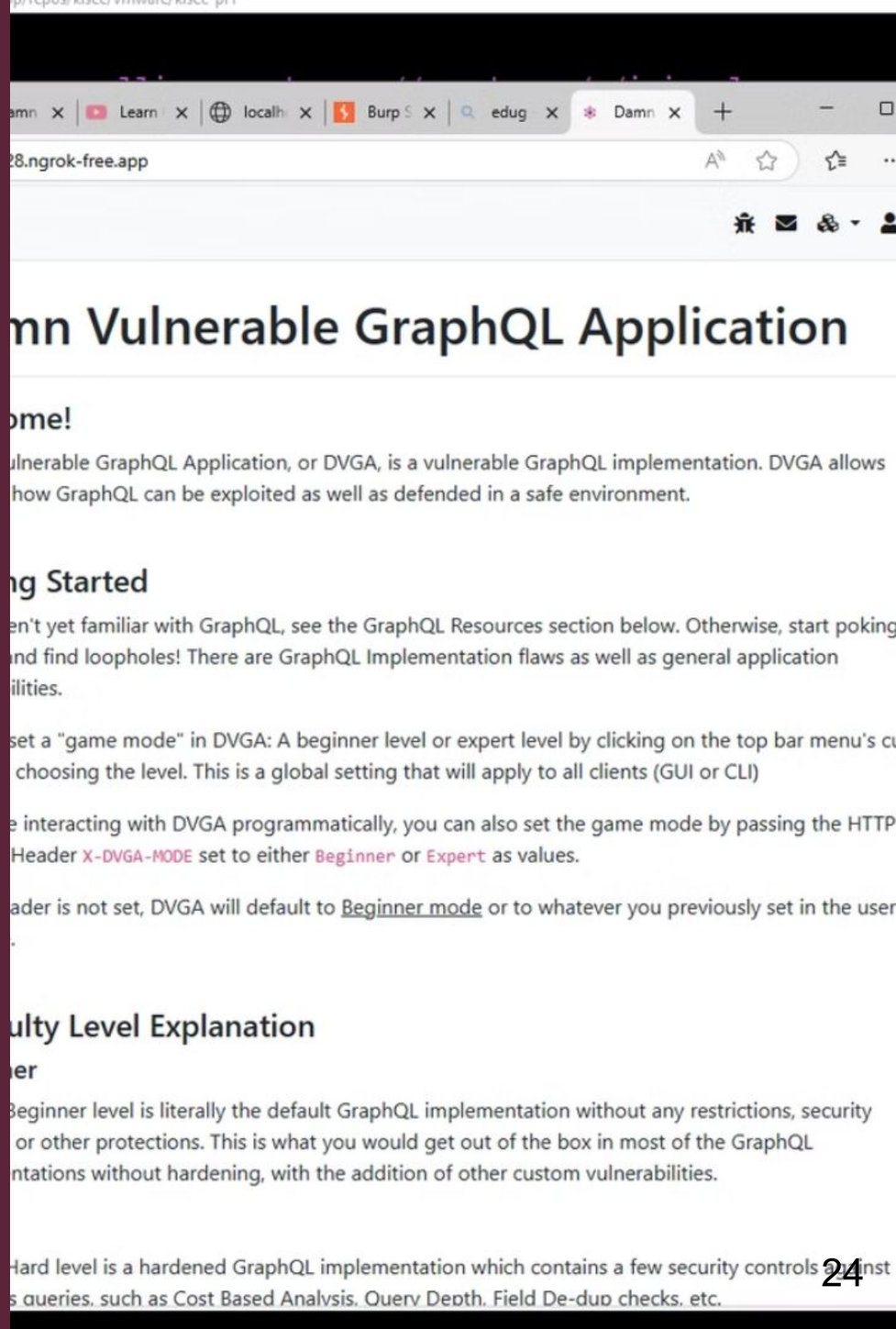
- Poor sanitization
- Information leakage
- Unnecessary data exposure

## SSL Certificate Issues

- Misconfigured HTTPS
- Self-signed certificates
- Domain correlation possible

## IPv6 Vulnerabilities

- Alternative attack paths
- Less monitored channel
- Security bypass potential



# Endpoint Discovery

## Curl Command Testing

Test __typename query via direct POST request

## Path Enumeration

Use ffuf with GraphQL-specific wordlists

## Response Filtering

Filter status codes: 200, 401, 403

## SecList Resources

Utilize specialized GraphQL wordlists

```
kisec@DESKTOP-1C4EIQM MINGW64 ~
$ curl https://ebb6-182-208-176-28.ngrok-free.app/v2/graphql
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manuall
y please check your spelling and try again.</p>

kisec@DESKTOP-1C4EIQM MINGW64 ~
$ curl https://ebb6-182-208-176-28.ngrok-free.app/graphql
{"errors":[{"message":"Must provide query string."}]}
kisec@DESKTOP-1C4EIQM MINGW64 ~
$ curl -X POST https://ebb6-182-208-176-28.ngrok-free.app/graphql \
     -H "Content-Type: application/json" \
     --data '{"query":"{ __typename }"}'
{"data":{"__typename":"Query"}}
```

```
v1.1.0
________________________________________________

 :: Method           : GET
 :: URL              : https://3cbe-182-208-176-28.ngrok-free.app/FUZZ
 :: Wordlist         : FUZZ: graphql-mini.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
________________________________________________

:: Progress: [14/14] :: Job [1/1] :: 14 req/sec :: Duration: [0:00:01] :: Errors: 0 ::
kisec@DESKTOP-1C4EIQM:~$ ffuf -w graphql-mini.txt -u https://3cbe-182-208-176-28.ngrok-free.app/FUZZ -mc 200,403,401 -fc 404 -v

v1.1.0
________________________________________________

 :: Method           : GET
 :: URL              : https://3cbe-182-208-176-28.ngrok-free.app/FUZZ
 :: Wordlist         : FUZZ: graphql-mini.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,403,401
 :: Filter           : Response status: 404
________________________________________________

:: Progress: [14/14] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
kisec@DESKTOP-1C4EIQM:~$
```

# Critical Findings

### Introspection Enabled

Complete schema exposure

### Sensitive Types Exposed

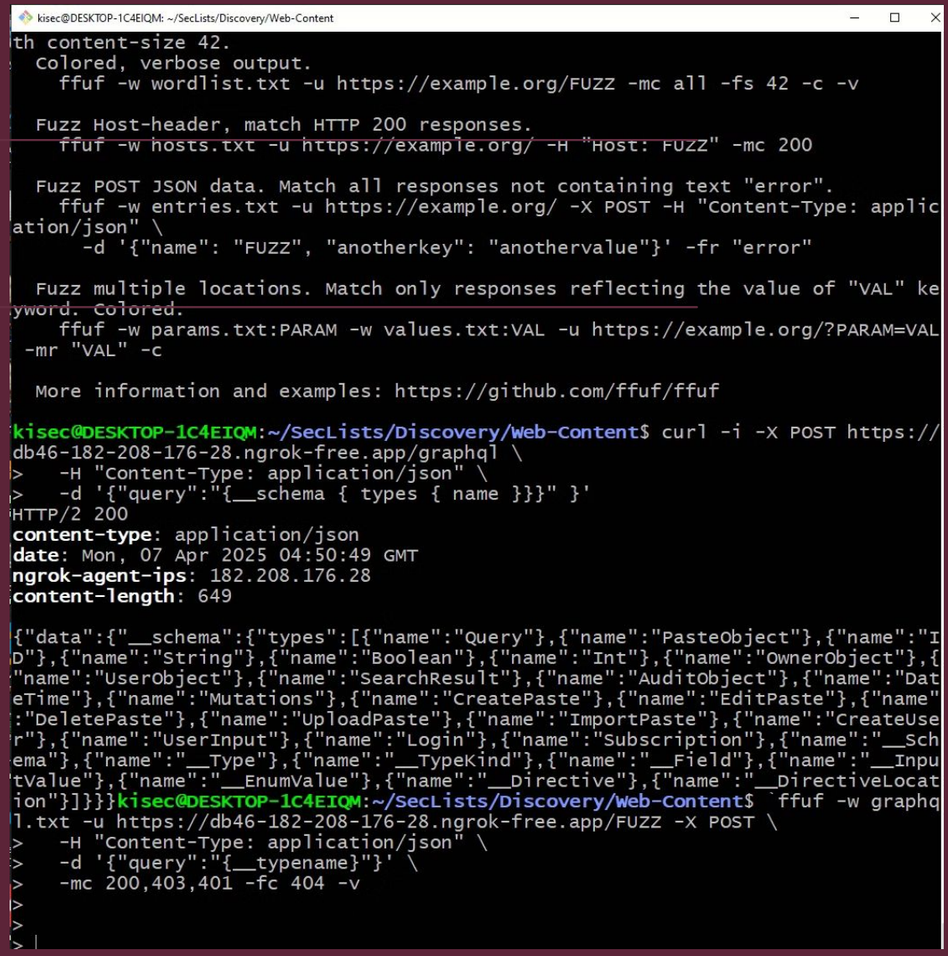
PasteObject, UserObject, AuditObject

### Dangerous Mutations

createUser, login, createPaste, editPaste

### Missing Protections

No rate limiting or auth checks






Made with Gamma

# REPORTING

https://github.com/aiegoo/kisec-present/Reporting.md

## 🛡️ Security Assessment Report

### GraphQL Reconnaissance & Attack Surface Enumeration

**Project:** KISEC Project 1
**Author:** @aiegoo
**Date:** April 2025

---

### 1. Executive Summary

This report summarizes the findings of a reconnaissance and enumeration assessment conducted on a GraphQL endpoint deployed on port `5013`. Through the use of open-source security tools and custom payloads, we discovered several security risks associated with an improperly secured GraphQL API and WebSocket-based subscription endpoint.

---

### 2. Objectives

- Identify exposed GraphQL endpoints
- Perform schema introspection to enumerate data types and operations
- Detect WebSocket upgrade mechanisms tied to GraphQL subscriptions
- Determine if access control or query restrictions are enforced
- Map the attack surface based on response behavior



Made with Gamma

# Source Code & Dependency Analysis - **Next Agenda**

## Static Application Security Testing

Run Bandit against Python codebase to identify security issues in source code. The recursive scan examines all Python files in the application directory, flagging potential security vulnerabilities based on known patterns:

```
bandit -r .
```

Review flagged issues for severity and exploitability in the GraphQL implementation.

## Pattern-Based Vulnerability Detection

Use Semgrep to identify security weaknesses based on code patterns specific to GraphQL implementations:

```
semgrep scan .
```

Focus on authorization bypasses, injection vulnerabilities, and information disclosure risks common in GraphQL applications.

## GraphQL Schema Analysis

Manually inspect the GraphQL schema for sensitive operations, excessive permissions, or information leakage. Analyze resolvers for proper authorization checks and input sanitization.

Source code analysis provides deeper insights into potential vulnerabilities that might not be detectable through external scanning. By examining the application's internal structure, we can identify architectural weaknesses and implementation flaws.



Made with Gamma