GraphQLmap v2.0 — Simple Usage Guide

TL;DR — Just Copy-Paste These

```bash
# Install
cd GraphQLmap-Updated
pip install -r requirements.txt

# Run automatic scan (does EVERYTHING for you)
python3 bin/graphqlmap -u https://target.com/graphql --auto

# That's it. Results are saved to a folder automatically.
```

Your 3 Questions Answered

1. How do I pass the URL?

Use the `-u` flag. The URL is the GraphQL endpoint:

```bash
python3 bin/graphqlmap -u https://target.com/graphql --auto
```

Common GraphQL endpoint paths:

- `https://target.com/graphql`
- `https://target.com/api/graphql`
- `https://target.com/v1/graphql`
- `https://target.com/graphiql`

If you don't know the exact path, just give the base URL and add `--detect`:

```bash
python3 bin/graphqlmap -u https://target.com --auto --detect
```

The tool will automatically scan 23 common paths to find it.

---

2. How do I pass the body (the actual GraphQL query)?

Use the `--body` (or `-b`) flag. The body is the GraphQL query/mutation you want to test:

Bash

```
# Simple query
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --body '{users{id,name,email}}'

# Query with arguments
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --body '{user(id: 1 ){name, email, role}}'

# Mutation
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --body 'mutation{login(username:"admin", password:"test" ){token}}'

# From a file (for complex queries)
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --body-file my_query.graphql
```

What is the body? It's the same thing you'd type in GraphiQL, Altair, or Burp Suite. For example, if you normally send this POST request:

JSON

```
POST /graphql HTTP/1.1
Content-Type: application/json

{"query": "{users{id,name,email}}"}
```

Then the `--body` value is just: `{users{id,name,email}}`

The tool wraps it in the JSON `{"query": "..."}` format automatically.

---

3. What does `--auto` do?

It runs the **entire attack chain automatically** in this order:

Plain Text

```
Phase 1:  Verify the GraphQL endpoint is alive
Phase 2:  Fingerprint the server engine (Apollo? Hasura? etc. )
Phase 3:  Try introspection to dump the full schema
Phase 4:  If introspection is blocked → try 12 bypass techniques
Phase 5:  If still blocked → brute-force field names from wordlist
```

```
Phase 6:  Execute your custom query (if you gave --body)
Phase 7:  Test for SQL injection & NoSQL injection
Phase 8:  Test batching & rate limit bypass (alias attacks)
Phase 9:  Test for CSRF vulnerabilities
Phase 10: Test DoS protections (depth, alias, directive overloading)
Phase 11: Test WebSocket subscriptions
Phase 12: Generate a full report with all findings
```

All results are saved to a folder: `graphqlmap_results_YYYYMMDD_HHMMSS/`

Complete Examples

Example 1: Basic auto scan (simplest possible)

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto
```

Example 2: With authentication token

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --headers '{"Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoiYWRtaW4
```

Example 3: With a specific query to test

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --body '{users{id,name,email,role,password}}'
```

Example 4: With cookie-based auth

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --headers '{"Cookie": "session=abc123def456"}'
```

Example 5: Through Burp Suite proxy

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --proxy http://127.0.0.1:8080
```

Example 6: Skip DoS tests (safer for production)

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto --skip-dos
```

Example 7: Custom wordlist for field brute-force

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --wordlist /path/to/my_wordlist.txt
```

Example 8: Save results to specific folder

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto \
  --output /path/to/results/
```

Example 9: Use GET method instead of POST

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql --auto --method GET
```

Example 10: Interactive mode (manual, command by command)

Bash

```
python3 bin/graphqlmap -u https://target.com/graphql
# Then type commands like:
#   fingerprint
#   dump_via_introspection
#   introspection_bypass
```

```
#   nosqli
#   help    (to see all commands )
```

Understanding the URL vs Body

Here's a visual breakdown:

Plain Text

```
┌──────────────────────────────────────────────────────────────────────┐
│                          HTTP REQUEST                                │
├──────────────────────────────────────────────────────────────────────┤
│                                                                      │
│  POST https://target.com/graphql    ← This is the -u URL             │
│  Content-Type: application/json                                      │
│  Authorization: Bearer TOKEN        ← This is --headers              │
│                                                                      │
│  {                                                                   │
│    "query": "{users{id,name}}"      ← This is the --body             │
│  }                                                                   │
│                                                                      │
└──────────────────────────────────────────────────────────────────────┘
```

Flag	What it is	Example
`-u`	The GraphQL endpoint URL	`https://target.com/graphql`
`--body`	The GraphQL query/mutation	`{users{id,name,email}}`
`--headers`	HTTP headers (auth, cookies)	`{"Authorization":"Bearer TOKEN"}`
`--proxy`	Proxy to route traffic through	`http://127.0.0.1:8080`
`--auto`	Run everything automatically	(no value needed)

Output Files

After `--auto` scan, you'll find these in the results folder:

File	Contents

`report.md`	Human-readable vulnerability report
`scan_results.json`	Machine-readable full results
`schema_full.json`	Complete GraphQL schema (if introspection worked)
`schema_simplified.json`	Simplified schema (types, queries, mutations)
`discovered_fields.json`	Fields found via brute-force (if introspection was blocked)
`custom_query_result.json`	Response from your `--body` query
`csrf_poc_get.html`	CSRF proof-of-concept (if vulnerable)
`csrf_poc_form.html`	CSRF form-based proof-of-concept (if vulnerable)

Quick Reference Card

Plain Text

```
╔═══════════════════════════════════════════════════════════════╗
║  GRAPHQLMAP v2.0 - QUICK REFERENCE                            ║
╠═══════════════════════════════════════════════════════════════╣
║                                                               ║
║  AUTOMATIC:  python3 bin/graphqlmap -u URL --auto             ║
║  + QUERY:    python3 bin/graphqlmap -u URL --auto -b 'QUERY'  ║
║  + AUTH:     ... --headers '{"Authorization":"Bearer T"}'     ║
║  + PROXY:    ... --proxy http://127.0.0.1:8080                ║
║  + SAFE:     ... --skip-dos                                   ║
║  MANUAL:     python3 bin/graphqlmap -u URL                    ║
║                                                               ║
╚═══════════════════════════════════════════════════════════════╝
```