ВВЕДЕНИЕ В
КИБЕРБЕЗ

Больше киберугроз

NoSQ
L

NoSQL

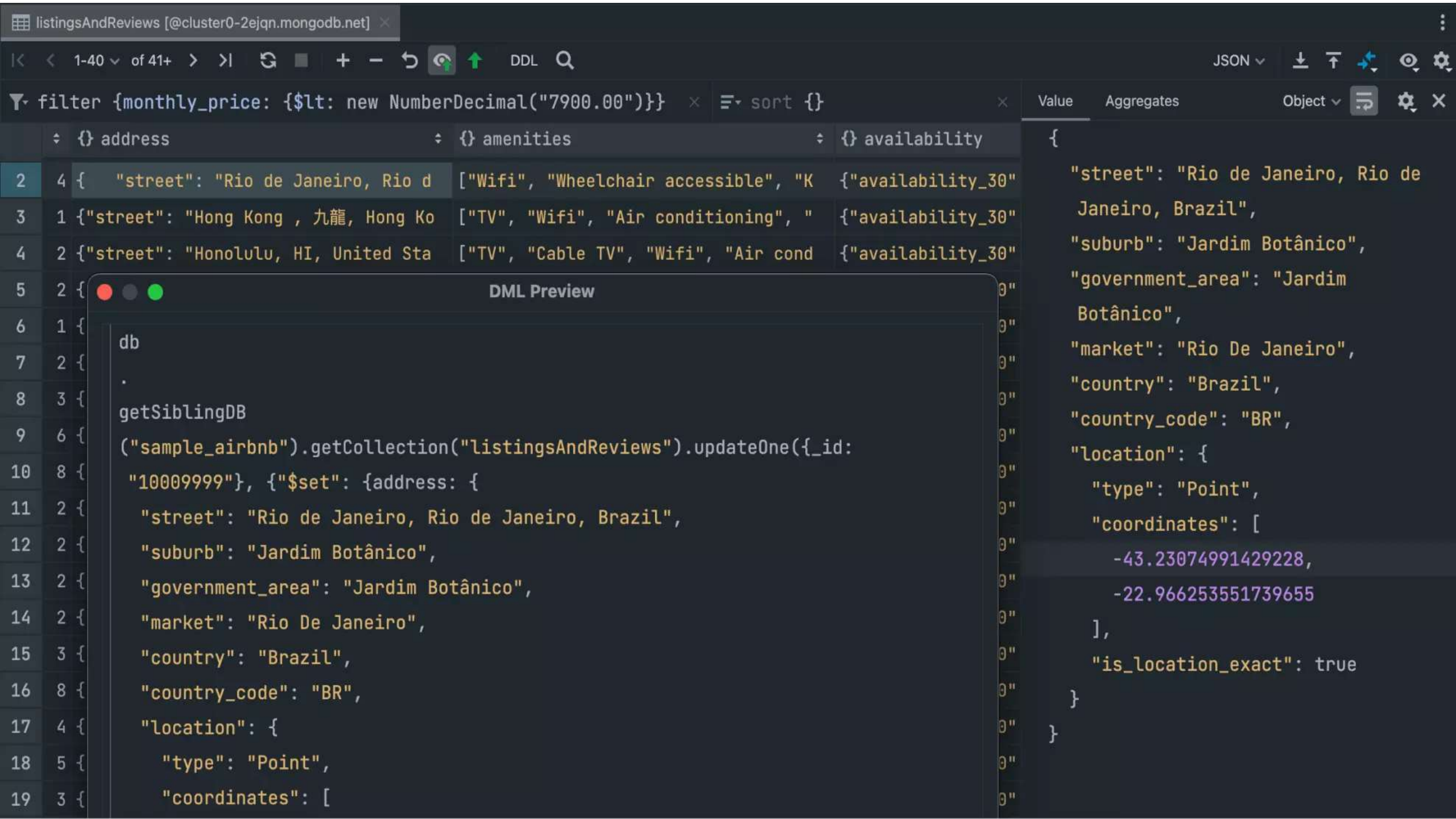
В отличие от своего брата **SQL, NoSQL (Not Only SQL)** - это тип БД, позволяющей хранить данные в различных форматах: ключ-значение, графы, документы, колоночные хранилища и многие другие...

NoSQL

Популярные виды **NoSQL**:

- **Ключ-значение (Key-Value)**: Хранят данные в виде пар ключ-значение, где ключ уникален и используется для поиска значения
- **Документные базы данных**: Хранят данные в виде документов, обычно в формате JSON, BSON или XML.
- **Колоночные базы данных**: Данные хранятся в виде колонок, что делает их более эффективными для чтения больших объемов данных.
- **Графовые базы данных**: Хранят данные в виде графов, где узлы представляют объекты, а рёбра – связи между ними.

Feature	SQL and NoSQL database difference	
	SQL	NoSQL
Data model	Structured	Unstructured or semi-structured
Scalability	Vertical scaling	Horizontal scaling
Performance	Complex queries can be slow	Fast read and write performance
ACID compliance	Yes	No or partial
Schema	Rigid	Flexible
Language	SQL (Structured Query Language)	JSON (JavaScript Object Notation), XML, YAML, or binary schema


listingsAndReviews [@cluster0-2ejqn.mongodb.net]
1-40 of 41+
DDL
JSON
filter {monthly_price: {$lt: new NumberDecimal("7900.00")}}
sort {}
Value
Aggregates
Object
{} address
{} amenities
{} availability
DML Preview
db
.
getSiblingDB
("sample_airbnb").getCollection("listingsAndReviews").updateOne({_id:
"10009999"}, {"$set": {address: {
"street": "Rio de Janeiro, Rio de Janeiro, Brazil",
"suburb": "Jardim Botânico",
"government_area": "Jardim Botânico",
"market": "Rio De Janeiro",
"country": "Brazil",
"country_code": "BR",
"location": {
"type": "Point",
"coordinates": [
-43.23074991429228,
-22.966253551739655
],
"is_location_exact": true
}
}

Проблемы безопасности в NoSQL

NoSQL

Несмотря на *множество преимуществ*, **NoSQL** базы данных могут иметь свои *уязвимости и недостатки*, особенно если они неправильно настроены. Например:

NoSQL

Несмотря на *множество преимуществ*, **NoSQL** базы данных могут иметь свои *уязвимости и недостатки*, особенно если они неправильно настроены. Например:

1. **NoSQL Injection** – подобно SQL-инъекции, NoSQL-инъекция возникает, когда данные от пользователя вставляются в запрос **без предварительной валидации**

NoSQL

Несмотря на *множество преимуществ*, **NoSQL** базы данных могут иметь свои *уязвимости и недостатки*, особенно если они неправильно настроены. Например:

1. **NoSQL Injection** – подобно SQL-инъекции, NoSQL-инъекция возникает, когда данные от пользователя вставляются в запрос **без предварительной валидации**
2. **Insecure Deserialization** – при работе с NoSQL базами также может быть опасность **десериализации пользовательских данных** без должной проверки

NoSQL

Например, **MongoDB** использует **коллекции** для хранения документов.
Основные запросы будут выглядеть следующим образом:

- Вставка документа

```
db.users.insertOne({
  name: “Kurt Cobain",
  age: 30,
  email: “nirvana@nrv.com"
});
```

- Вставка нескольких документов

```
 db.users.insertMany([
   { name: “Kamaz", age: 25,
email: “kumazz@huhheh.com" },
   { name: “Sasun", age: 28,
email: “sasunbek@uzb.uzb" }
 ]);
```

NoSQL

- Получение всех документов из коллекции

```
db.users.find();
```

- Вставка конкретного дока

```
db.users.findOne({ name: "John Doe" });
```

- Поиск с условием

```
db.users.find({ age: { $gte: 30 } });
```

NoSQL

» Поиск с составным условием

```
db.users.find({
  $and: [
    { age: { $gte: 30 }},
    { status: "active" }
  ]
});
```

=

» Поиск с составным условием

```
db.users.find({
    { age: { $gte: 30 } },
    { status: "active" }
});
```

NoSQL

- Получение всех документов из коллекции

```
db.users.find();
```

- Вставка конкретного дока

```
db.users.findOne({ name: "John Doe" });
```

- Поиск с условием

```
db.users.find({ age: { $gte: 30 } });
```

NoSQL

- Обновление документа

```
db.users.updateOne(
  { name: "Kurt Cobain" },
  { $set { age: 30 }},
   email: "nirvana@nrv.com"
});
```

- Обновление нескольких документов

```
 db.users.updateMany(
   { age: { $gte: 30 } },
   { $set: { name: "S.K.U.F" } }
);
```

NoSQL

- Обновление документа

```
db.users.deleteOne(
  { name: "Kurt Cobain" }
);
```

- Обновление нескольких документов

```
db.users.deleteMany(
  { age: { $lte: 5 } }
);
```

NoSQLi

Для тестирования на NoSQLi мы можем воспользоваться подстановкой неочевидных значений в запрос

Например:

```
'"`{;$Foo} $Foo \xYZ
```

Такой запрос скорее всего сломает логику приложения:

```
https://insecure-website.com/product/lookup?category='%22%60%7b%0d%0a%3b%24Foo%7d%0d%0a%24Foo%20%5cxYZ%00
```

NoSQLi

Рассмотрим, как выглядит NoSQL инъекция. Найдем чем этот тип БД схож со своим «старшим братом» ;)

NoSQLi syntax

Рассмотрим, как выглядит NoSQL инъекция. Найдем чем этот тип БД схож со своим «старшим братом» ;)

Допустим есть запрос:

```
this.category == 'fizzy' && this.released == 1
```

NoSQLi syntax

Рассмотрим, как выглядит NoSQL инъекция. Найдем чем этот тип БД схож со своим «старшим братом» ;)

Допустим есть запрос:

```
this.category == 'fizzy' && this.released == 1
```

Тогда эксплойт:

```
this.category == 'fizzy‘ || ‘1’==‘1’ && this.released == 1
```

NoSQLi syntax

Рассмотрим, как выглядит NoSQL инъекция. Найдем чем этот тип БД схож со своим «старшим братом» ;)

Допустим есть запрос:

```
this.category == 'fizzy' && this.released == 1
```

Тогда эксплойт:

```
this.category == 'fizzy‘ || ‘1’==‘1’ && this.released == 1
```

```
this.category == 'fizzy’%00 ‘ && this.released == 1
```

NoSQLi param

Рассмотрим, как выглядит NoSQL инъекция. Найдем чем этот тип БД схож со своим «старшим братом» ;)

```
 db.users.find({ username: req.body.username, password:
req.body.password })
```

Злоумышленник может отправить такой запрос:

```
{ "username": { "$ne": null }, "password": { "$ne": null } }
```

Какой импакт атаки из примера?

NoSQLi param

```
{ "username": { "$ne": null }, "password": { "$ne": null } }
```

```
{
  "_id": ObjectId("someid"),
  "username": “Petya_Podvohov",
  "password": "mypassword123"
},
{
  "_id": ObjectId("anotherid"),
  "username": “AMOGUSDRIP",
  "password": “42bratuha!"
}, ……
```

NoSQLi param

Что вернет запрос?

```
db.users.find({
  $and: [
    { age: { $gte: 18, $lt: 60 } },
    { username: { $regex: "^A" } }  // UserName starts w 'A'
  ]
});
```

NoSQLi param

Мы можем так же сделать эксфильтрацию данных:

```
 db.users.find({ “username”:”NikokadoAvokado”,
  "password": { $regex: "^p.*" }  // Пароль начинается с "p"
});
```

NoSQLi param

Мы можем так же сделать эксфильтрацию данных:

```
 db.users.find({ “username”:”NikokadoAvokado”,
  "password": { $regex: "^p.*" }  // Пароль начинается с "p"
});
```

```
 db.users.find({ “username”:”NikokadoAvokado”,
  "password": { $regex: "^pa.*" }  // По мере брута узнаем символы
});
```

NoSQLi param

Мы можем так же сделать эксфильтрацию данных:

```
 db.users.find({ “username”:”NikokadoAvokado”,
  "password": { $regex: "^p.*" }  // Пароль начинается с "p"
});
```

```
 db.users.find({ “username”:”NikokadoAvokado”,
  "password": { $regex: "^pa.*" }  // По мере брута узнаем символы
});
```

```
 db.users.find({ “username”:”NikokadoAvokado”,
  "password": { $regex: "^pas.*" }  // По мере брута узнаем символы
});
```

NoSQLi param

```
 db.users.find({
  "password": { $not: /[^a-zA-Z0-9]/ }  // Пароль только из
букв/цифр
});
```

NoSQLi practice

https://portswigger.net/web-security/nosql-injection#nosql-syntax-injection

Path Traversal

Path Traversal

Уязвимость, которая позволяет злоумышленнику **обойти ограничения** на **доступ к файлам** в операционной системе

Атака позволяет получить доступ к файлам, которые находятся **вне разрешённой директории**

```php
<?php

$file = $_GET['file'];

include('/var/www/images/' . $file);

?>
```


<?php
$file = $_GET['file'];
include('/var/www/images/' . $file);
?>

Path Traversal

Почему это существует?

Атака использует специальные символы в путях файлов, такие как **../**, чтобы перемещаться вверх по файловой системе.

Если система **неправильно обрабатывает пути**, злоумышленник может **прочитать** или **изменить файлы**, находящиеся вне области доступа. Говорил же я вам, что **линуху учить надо…**

../../../../../etc/passwd

Path Traversal

Допустим веб-приложение позволяет пользователям загружать файлы или просматривать их содержимое, запрашивая путь:

```
/files/view?file=profile.txt
```

Попробуем поменять **profile.txt** на что-то другое:

```
/files/view?file=../../../../../../etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

Импакт?

Как защищаться?

Path Traversal

Как защититься, говорите?

1. Применение фильтрации и экранирования специальных символов
2. Проверка, что путь находится в пределах разрешённых директорий
3. Использование абсолютных путей

```
function sanitizeInput(input) {
    return input.replace(/(\.\.\/|\.\.\\|\/|\\)/g, '');
}
```

https://portswigger.net/web-security/file-path-traversal

Information Disclosure

Inf Disc

Приложение или система непреднамеренно или несанкционированно раскрывает **конфиденциальную информацию**, которая может быть использована злоумышленниками для *дальнейших атак*.

Есть множество возможных случаев раскрытия инфы. Давайте посмотрим наиболее популярные

Inf Disc

Утечка данных в заголовках HTTP:

- X-Powered-By: указывает используемую платформу (например PHP)
- Server: может показывать тип веб-сервера (например, Apache или Nginx)
- WWW-Authenticate: инфа о методах аутентификации

Утечка данных в теле ответа:

- Стек ошибок или трассировки ошибок
- Ключи API в ответах серва

Утечка данных через файлы конфигурации (мы сможем просмотреть конфиги)

Утечка данных через лог-файлы (например, секреты)

Импакт?

Inf Disc

- Упрощение атак (путем раскрытия инфы)
- Повышение привилегий
- Потеря доверия и репутации
- Юридические последствия

Do you know
what time it is?

Do you know what time it is?

Practice time!

https://portswigger.net/web-security/information-disclosure/exploiting#common-sources-of-information-disclosure