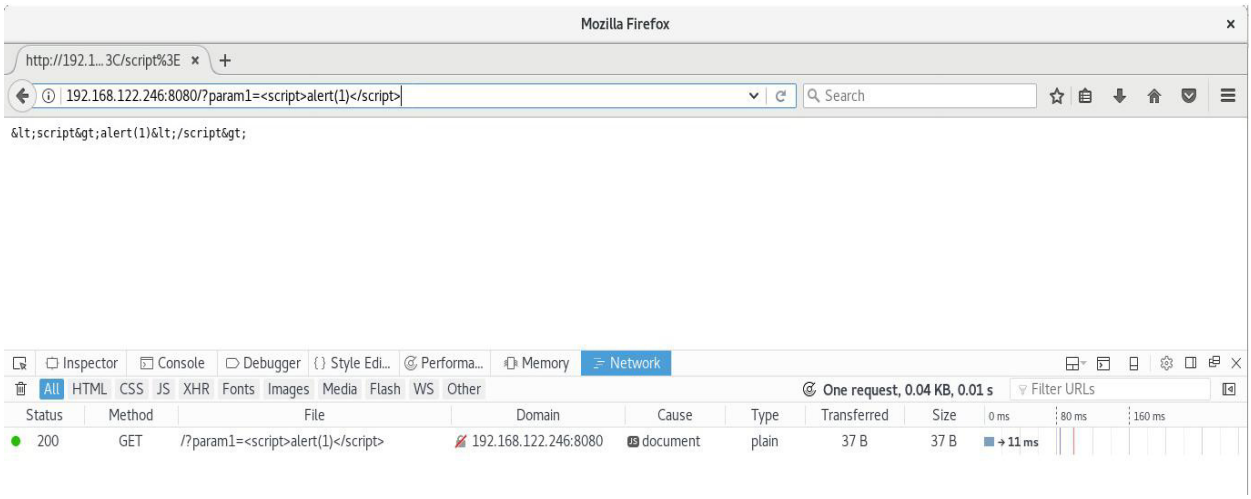

7.12.2 SQL Injection

Come riconoscerla

L'SQL Injection nasce dalla mancata/non corretta codifica dei dati di input/output. Partendo dalla query di esempio riportata di seguito:

```
ctx := context.Background()
customerId := r.URL.Query().Get("id")
query := "SELECT number, expireDate, cvv FROM creditcards WHERE customerId = " +
customerId
row, _ := db.QueryContext(ctx, query)
```

Quando viene fornito un customerId valido, la query restituisce l'elenco delle carte di credito del cliente.

Tuttavia, se customerId non è un valore, ma una stringa (concatenazione di diversi valori/simboli) come nell'esempio che segue:

```
SELECT number, expireDate, cvv FROM creditcards WHERE customerId = 1 OR 1=1
```

La query restituirebbe (a meno di opportune verifiche dei dati immessi in input) tutti i record della tabella relativamente a tutti i clienti censiti poiché la condizione 1 = 1 sarà 'truè per qualsiasi record.

Come difendersi

Impostare i placeholder:

```
ctx := context.Background()
customerId := r.URL.Query().Get("id")
query := "SELECT number, expireDate, cvv FROM creditcards WHERE customerId = ?"
stmt, _ := db.QueryContext(ctx, query, customerId)
```

La sintassi è specifica:

MySQL	PostgreSQL	Oracle
`WHERE col = ?`	`WHERE col = $1`	`WHERE col = :col`
`VALUES(?, ?, ?)`	`VALUES($1, $2, $3)`	`VALUES(:val1, :val2, :val3)`