Talos Vulnerability Report

TALOS-2020-1091

ERPNext frappe.desk.reportview.get SQL injection vulnerability

AUGUST 18, 2020

CVE NUMBER

CVE-2020-6145

SUMMARY

An SQL injection vulnerability exists in the frappe.desk.reportview.get functionality of ERPNext 11.1.38. A specially crafted HTTP request can cause an SQL injection. An attacker can make an authenticated HTTP request to trigger this vulnerability.

CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

ERPNext 11.1.38

PRODUCT URLS

ERPNext - https://erpnext.com/

CVSSV3 SCORE

6.4 - CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N

CWE

CWE-89 - Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

DETAILS

ERPNext is a free and open-source integrated Enterprise Resource Planning software developed by Frappé Technologies Pvt. Ltd. and is built on MariaDB database system using a Python based server-side framework. ERPNext is a generic ERP software used by manufacturers, distributors and services companies.

The `field` parameter in the `frappe.desk.reportview.get` handler is vulnerable to SQL injection.

Below is an example post that will trigger the vulnerability:

```
POST / HTTP/1.1
Host: [ip]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
Accept: application/json
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Frappe-CSRF-Token: fd9dfd2d0ac5e0ccf137d6de7d9c5b0d45fb828d636fe379efc121ce
X-Requested-With: XMLHttpRequest
Content-Length: 862
Origin: http://[ip]
DNT: 1
Connection: close
Referer: http://[ip]/desk
Cookie: user_image=; user_id=[User]; system_user=yes; full_name=[User]; sid=9ddfcdd569616e3f62f147517220861e1d12edeeaff2db5ffd4a6d10;
io=gJKBCzo-R8XbxV-nAAAD

doctype=Lead&fields=%5B%22%60tabLead%60.%60name%60%22%2C%22%60tabLead%60.%60owner%60%22%2C%22%60tabLead%60.%60creation%60%22%2C%22%60tabLead
%60.%60modified%60%22%2C%22%60tabLead%60.%60modified_by%60%22%2C%22%60tabLead%60.%60_user_tags%60%22%2C%22%60tabLead%60.%60_comments%60%22%2
C%22%60tabLead%60.%60_assign%60%22%2C%22%60tabLead%60.%60_liked_by%60%22%2C%22%60tabLead%60.%60docstatus%60%22%2C%22%60tabLead%60.%60parent%
60%22%2C%22%60tabLead%60.%60parenttype%60%22%2C%22%60tabLead%60.%60parentfield%60%22%2C%22%60tabLead%60.%60idx%60%22%2C%22%60tabLead%60.%60c
ompany_name%60%22%2C%22%60tabLead%60.%60status%60%22%2C%22%60tabLead%60.%60lead_owner%60%22%2C%22%60tabLead%60.%60lead_name%60%22%2C%22%60ta
bLead%60.%60image%60%22%5D[SQLINJECTION]&filters=%5B%5D&order_by=%60tabLead%60.%60modified%60+desc&start=0&page_length=20&with_comment_count
=true&cmd=frappe.desk.reportview.get
```

The parameter get passed to the `frappe/frappe-bench/apps/frappe/frappe/desk/reportview.py` source file, where first the function `get` is called to parse all structures in the request, along with JSON queries, as seen on line 22, before separate `compress` and `execute` functions are called as seen on the same line:

```
 17 @frappe.whitelist()
 18 @frappe.read_only()
 19 def get():
 20         args = get_form_params()
 21
 22         data = compress(execute(**args), args = args)
 23
 24         return data
 25
 26 def execute(doctype, *args, **kwargs):
 27         return DatabaseQuery(doctype).execute(*args, **kwargs)
```

Function ‘execute’ will then simply call ‘DatabaseQuery’ function in ‘frappe/frappe/model/db_query.py’ where on line 93, function to create specific SQL query is called:

```
 89
 90                if query:
 91                        result = self.run_custom_query(query)
 92                else:
 93                        result = self.build_and_run()
```

Before final execution in the ‘frappe/frappe/model/db_query.py’ source file as seen below, where the original parameter is passed to create a specific SQL Query::

```
104         def build_and_run(self):
105                 args = self.prepare_args()
106                 args.limit = self.add_limit()
107
108                 if args.conditions:
109                         args.conditions = "where " + args.conditions
110
111                 if self.distinct:
112                         args.fields = 'distinct ' + args.fields
113
114                 query = """select %(fields)s from %(tables)s %(conditions)s
115                         %(group_by)s %(order_by)s %(limit)s""" % args
```

TIMELINE

2020-06-09 - Vendor Disclosure

2020-08-18 - Public Release

CREDIT

Discovered by Yuri Kramarz of Cisco Talos.
