MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY

CSIRT.SK

Nette Framework Vulnerability Permits SQL Injection

Created by: CSIRT.SK
Ministerstvo investícií, regionálneho rozvoja a informatizácie SR
Pribinova 25
811 09 Bratislava

Date of creation: October 2024

TLP: Clear

Security analysts of CSIRT.SK discovered a potential vulnerability in web framework Nette. The security flaw allows to perform SQL injection attacks.

The vulnerability is found in the Database library and stems from the absence of sanitization of user input, which is passed to the *where($by)* function. This method allows filtering database query results by field, where keys represent column names and their values represent required values. Problematic are keys of variables pairs *{"key":"value"}*.

The attacker can perform an SQL injection attack by passing a key of a variable strait in a URL in browser, or into an application input, based on a specific case.

If an attacker can send a PHP array in a form of *{"SQLi payload"=>"value"}* to the method *where*, where the ***SQLi payload*** represents a user input, the **payload** will be executed in the database. Thus the attacker can get access to sensitive information. Attack example:

```
http://localhost:8000/?id%3d1)+UNION+SELECT+2,version()+FROM+test+WHERE+(1=1
```

This way the following database query is be built in Nette Framework:

```
"SELECT id, name FROM test (WHERE ". $column . " = ?)"
```

By filling in the value for *column* parameter we get:

```
"SELECT id, name FROM test (WHERE ". "id=1) UNION SELECT 2,version() FROM test WHERE (1" . " = ?)"
```

Thus the final parametrized query would look like this:

```
"SELECT id, name FROM test (WHERE id=1) UNION SELECT 2,version() FROM test WHERE (1= ?)"
```

The vulnerability is present in a web application, if it uses Nette/Database library, and if the following code is used in Presenter:

```
$httpRequest = $this->getHttpRequest();

$filter = $httpRequest->getQuery();

$this->template->terms = $this->testFacade->findBy($filter);
```

At the same time Facade uses the *findBy* method defined by (it takes value from *where* function):

```
public function findBy(array $by)  {

   return $this->database->table($this->tableName)->where($by);

 }
```

Possible damages:

- **Information disclosure**

Recommendations:

If you implement the vulnerable function in your Nette Framework web application, we recommend additional sanitization of user inputs (key of the variable) within the best practices of secure development.